Documento de trabajo

A MATLAB Toolbox for reliable time series modeling and forecasting in State-Space

Jaime Terceiro
José Manuel Casals
Miguel Jerez
Gregorio R. Serrano
Sonia Sotoca

No. 0005 Junio 2000

ICAE

Instituto Complutense de Análisis Económico

UNIVERSIDAD COMPLUTENSE

ICAE

Instituto Complutense de Análisis Económico

UNIVERSIDAD COMPLUTENSE

FACULTAD DE ECONOMICAS

Campus de Somosaguas

28223 MADRID

Teléfono 91 394 26 11 - FAX 91 294 26 13

Internet: http://www.ucm.es/info/icae/

E-mail: icaesec@ccee.ucm.es

A MATLAB TOOLBOX FOR RELIABLE TIME SERIES MODELING AND FORECASTING IN STATE-SPACE

Jaime Terceiro
José Casals
Miguel Jerez†
Gregorio R. Serrano
Sonia Sotoca

Departamento de Fundamentos del Análisis Económico II
Facultad de CC. Económicas y Empresariales
Universidad Complutense de Madrid

ABSTRACT

Software reliability is a wide issue, depending not only on the use of stable implementations of well-reputed algorithms, but also on software design aspects. This philosophy is implemented in E^4, a MATLAB Toolbox which uses state-space methods to achieve both, flexibility and reliability. E^4 supports many standard formulations such as VARMAX, econometric models in structural form, transfer functions or general linear state-space models. These models are estimated by exact maximum-likelihood, under standard conditions, or in an extended framework that allows for measurement errors, missing data, vector GARCH errors and constraints on the parameters. Ready-to-use functions are provided for model specification, preliminary estimation by subspace methods, analytic computation of the likelihood gradient and information matrix, simulation, forecasting and signal extraction. The core algorithms have been optimized for stability and numerical accuracy. In this aspect, the use of a computational platform such as MATLAB guarantees computational accuracy, portability and consistency between different hardware-software platforms. Several examples illustrate the main features of the Toolbox.

Keywords: Software reliability, State-space models, Kalman filter, Simulation, Smoothing, MATLAB

JEL Classification: C32, C53, C87

† Corresponding author. Departamento de Fundamentos del Análisis Económico II. Universidad Complutense de Madrid. Campus de Somosaguas. 28223 Madrid (Spain). Phone: (+34) 913 94 23 61. Fax: (+34) 913 94 26 13. E-mail: mjerez@ccee.ucm.es.

Foreword

This article describes the capacities and use of a MATLAB Toolbox for Time Series Analysis called E^4. It can be downloaded from the URL:

http://www.ucm.es/info/icae/e4

The main materials provided at this address are:

1) The source code of E^4. It is included in a .zip file locked with the keyword "isf2000". It can be freely downloaded for academic use, at least until 1/1/01. We reserve the right to change this policy in the future, see the note below.

2) A complete Reference Manual, see Terceiro *et al.* (2000), including a detailed description of all the public E^4 functions and many examples.

3) Source code and data for several E^4 applications and analyses, including the examples in the Reference Manual, see point 2) above, and those in Section 7 of this paper.

4) Technical papers (including this one), either published or in draft, describing the operating details of many functions.

5) A "Beta Test Corner", where we post useful functions in different degrees of development and testing. It includes a preliminary documentation for these functions. Many of them will become part of a future "official" E^4 release.

6) Contact information about the members of the E^4 development team and the current Application Administrator. Questions and bug reports should be sent to the Application Administrator. We will try to answer as many as possible.

Conditions of use and disclaimers

The E^4 WEB page is now under beta testing. This means that the information included and its structure will change in the next months. Also, some links and options may not work adequately. By 7/31/00 we expect to have a fully operative page.

The current version of E^4 is stable and thoroughly tested, but only runs on MATLAB 4.x. During this summer we will update it to support both, MATLAB 4.x and 5.x releases. We found some numerical problems in MATLAB 5.1 and 5.2, so we advise upgrading to 5.3 before using our Toolbox.

Until 1/1/01, the Toolbox is licensed for academic use. This means that you can download the code, test it and use it both, for teaching and research purposes. If you consider that an acknowledgment is due, it will be enough to mention the use of E^4 and to provide the URL of our WEB page.

1. Introduction

This paper describes a MATLAB Toolbox for econometric modeling of time series. Its name, E^4, refers to the Spanish *Estimación de modelos Econométricos en Espacio de los Estados*, meaning "State-Space Estimation of Econometric Models."

E^4 makes up for the lack of PC software for estimating econometric models by exact maximum likelihood. The main model supported is a general state-space (SS) form with fixed-parameters. On this basis, the Toolbox supports also many standard formulations such as VARMAX (Vector AutoRegressive Moving Average with eXogenous variables), structural econometric models and single-output transfer functions. All these models can be estimated: a) by themselves or in composite formulations, b) unconstrained or subject to linear and/or nonlinear constraints on the parameters, and c) under standard conditions (*i.e.*. with full information and homoscedastic errors) or in an extended framework that allows for observation errors, missing data and vector GARCH (Generalized AutoRegressive Conditional Heteroscedastic) errors.

Statistical and econometric packages such as SAS, S-PLUS, SPSS, PcGive and STAMP have adequate "canned" options for standard time series modeling in a fast and friendly environment. Whereas this approach is adequate for many users, time series research often needs additional reliability and flexibility. E^4 may attend these special needs through several unique features:

1) *Implementation in source code running on MATLAB*, which means that:
   - It is embedded in a powerful programming environment, able to alleviate mundane (but important) chores as: generating calendar variables, estimating the same model for several sub-samples or importing and exporting data.
   - The Toolbox is modular and reusable. This has two immediate implications. First, specific functions can be discarded and easily replaced by code designed according to other requirements. Second, if a function is found interesting, it can be immediately integrated in other MATLAB application(s).
   - We developed the Toolbox in a Windows 98/Intel environment, but it is immediately portable to other platforms where MATLAB implementations are available, including DEC Alpha, HP 9000, IBM RS/6000, Linux, Macintosh (68000 and Power Mac) VMS and SUN.

2) *The Toolbox was designed bottom-up for research*, meaning that:
   - Econometric functions provide complete output of computational results - *e.g.*, the gaussian information matrix or the likelihood gradient on convergence - which can be used *ex-post* to implement special tests or any calculation not supported specifically.
   - The Toolbox architecture has been designed to satisfy the needs of a sophisticated user,

requiring accurate information about all its computational aspects and easy-to-use expansion capacities.

- The code has been carefully optimized for numerical accuracy and robustness.

3) *Important computations are done using the equivalent state-space formulation of the model.* This feature has several implications:

- SS methods allow for certain computations and analyses that would otherwise be impossible, such as modeling samples with missing data or observation errors. They also allow us to use many well-tested numerical procedures, coming mainly from the aerospace engineering world.
- The code performing complex and fragile computations - such as likelihood evaluation or forecasting - is concentrated in a single function, which performance can be exhaustively tested.
- It is well known that SS methods can be difficult to use directly even for seasoned researchers. However, E^4 users do not need to understand nor handle the SS formulation, as the library includes many interface functions that manage the necessary conversions from/to the conventional representation of the model.
- Relying on a core of SS computational functions, surrounded by interface functions, make the Toolbox easy to extend, as implementing a new model which has a SS equivalent representation only requires coding a new interface routine or what we call a "user function", see Terceiro *et al.* (2000).

These features of E^4 have pros and cons. Specifically, running in a pseudo-compiling environment and emphasis on accuracy implies slow performance. We do not see this as a serious inconvenience, as econometric modeling seldom requires real-time feedback. However, speedier computations can be obtained easily by tuning some system parameters, or with some effort by translating critical functions to a lower level language, such as C, or generating MEX files from the source code, see MATLAB (1992) and MATLAB (1996). On the other hand, this emphasis on accuracy and robustness has many advantages for the researcher, see McCullough and Vinod (1999).

In Section 2 we introduce SS representations used by E^4. Section 3 discusses the formulation of several standard models such as the structural econometric model, VARMAX, and transfer functions. It also documents the functions provided by E^4 to define them.

Sections 4 and 5 discuss different cases of model combination. First, several simple models can be combined into a single formulation, called a "composite model". This option, documented in Section 4, allows one to deal with special situations such as unit roots, observation errors or multiple seasonal factors. Section 5 concentrates in the combination of a simple model for the (conditional) mean of the time series with a conditional heteroscedastic model (GARCH) for the errors.

Section 6 describes briefly the state-space algorithms employed for likelihood computation, forecasting, smoothing and simulation. These procedures handle transparently missing values and initial conditions of Kalman filter. Complete technical details are provided in several cited books and articles.

Section 7 illustrates the use of E^4 by means of complete input and output data for several applications to real data, including ARIMA estimation and forecasting; estimation and smoothing of an unobserved components model; estimation and diagnosis of a VARMA model and estimation of a bivariate model with vector GARCH errors.

Finally, Section 8 provides some concluding remarks and the Appendix summarizes the main E^4 functions.

2. State-Space models

State-space models provide an elegant and unified way to describe a wide class of linear stochastic processes, including as particular cases many standard formulations such as ARIMA and VARMAX. Also, the SS framework simplifies the analysis in nonstandard situations such as errors in variables, Terceiro (1990, Chapter 3), conditional heteroscedastic errors, Harvey, Ruiz and Sentana (1992), random coefficients, Swamy and Tavlas (1995), or structural time series models, Harvey (1989).

2.1 The state-space model with fixed coefficients

The fixed-coefficients SS formulation supported by E^4 is:

$$x_{t+1} = \Phi x_t + \Gamma u_t + E w_t \tag{1}$$

$$z_t = H x_t + D u_t + C v_t \tag{2}$$

where x_t is a $(n \times 1)$ vector of state variables, u_t is a $(r \times 1)$ vector of exogenous variables, z_t is a $(m \times 1)$ vector of observable variables, w_t and v_t are white noise processes such that:

$$E[w_t] = 0\,, \qquad E[v_t] = 0 \tag{3}$$

$$E\left[\begin{pmatrix} w_{t_1} \\ v_{t_2} \end{pmatrix} \begin{pmatrix} w_{t_1}^T & v_{t_2}^T \end{pmatrix}\right] = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \delta_{t_1 t_2} \tag{4}$$

being Q and R positive semi-definite matrices. Φ, Γ, E, H, D, C, Q, R and S are matrices constant

over time, see Terceiro (1990).

Example 2.1 (SS representation of standard econometric models). Any linear econometric model with fixed parameters can be represented in the SS form (1)-(4). Therefore, numerical or statistical procedures developed for the SS model can be applied to all the particular cases. This is the basic idea implemented in E[4].

Consider the well-known airline model:

$$\nabla\nabla_{12} y_t = (1 + \theta_1 B)(1 + \Theta_1 B^{12})\epsilon_t$$

where $\epsilon_t \sim iidN(0, \sigma_\epsilon^2)$ and y_t is log-transformed. Its equivalent representation in the form (1)-(4) is:

$$\boldsymbol{x}_{t+1} = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 \\ 0 & 0 & 1 & . & . & 0 \\ . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 \\ 0 & 0 & 0 & . & . & 0 \end{bmatrix} \boldsymbol{x}_t + \begin{bmatrix} \theta_1 \\ 0 \\ . \\ \Theta_1 \\ \theta_1\Theta_1 \end{bmatrix} \epsilon_t$$

$$y_t = \begin{bmatrix} 1 & 0 & 0 & . & . & 0 \end{bmatrix} \boldsymbol{x}_t + \epsilon_t$$

where $\boldsymbol{Q} = \boldsymbol{R} = \boldsymbol{S} = \sigma_\epsilon^2$ and the state dimension is $n = 12$. Literature propose several conversion procedures from the standard representation of a model to the equivalent SS form, see Terceiro (1990).

Example 2.2 (Unobserved components time series model). The ARIMA process in previous example corresponds approximately to the unobserved components model:

$$y_t = T_t + S_t + e_t$$

$$T_{t+1} = T_t + \Delta_t$$

$$\Delta_{t+1} = \Delta_t + \eta_t$$

$$(1 + B + B^2 + B^3 + \ldots + B^{11}) S_{t+1} = w_t$$

where T_t is the *trend component*, representing the long-term behavior of the series, S_t is the *seasonal component*, associated to persistent patterns repeated along a season, and e_t is an *irregular component*. The trend component follows a particular case of the "stochastic trend model", see Harvey (1989),

implying that the trend in $t+1$ is equal to the trend in t plus a random walk change of the trend, Δ_t. The seasonal component follows a monthly "dummy seasonal" model, where the sum of the seasonal components over a year is a random disturbance. The errors terms η_t, w_t and e_t are gaussian white noise processes with an instantaneous covariance matrix:

$$\mathrm{COV}\begin{bmatrix} \eta_t \\ w_t \\ e_t \end{bmatrix} = \begin{bmatrix} \sigma_\eta^2 & 0 & 0 \\ 0 & \sigma_w^2 & 0 \\ 0 & 0 & \sigma_e^2 \end{bmatrix}$$

For more details about these models, see Harvey (1989). This class of models is supported by E[4] through the use of formulation (1)-(4). In this case, the SS representation is:

$$\begin{bmatrix} T_{t+1} \\ \Delta_{t+1} \\ S_{t+1} \\ S_t \\ S_{t-1} \\ . \\ . \\ S_{t-9} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & . & . & 0 \\ 0 & 1 & 0 & 0 & 0 & . & . & 0 \\ 0 & 0 & -1 & -1 & -1 & . & . & -1 \\ 0 & 0 & 1 & 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & 1 & 0 & . & . & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & 0 & . & 1 & 0 \end{bmatrix} \begin{bmatrix} T_t \\ \Delta_t \\ S_t \\ S_{t-1} \\ S_{t-2} \\ . \\ . \\ S_{t-10} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ . & . \\ . & . \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \eta_t \\ w_t \end{bmatrix}$$

$$y_t = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & . & . & 0 \end{bmatrix} \begin{bmatrix} T_t \\ \Delta_t \\ S_t \\ S_{t-1} \\ S_{t-2} \\ . \\ . \\ S_{t-10} \end{bmatrix} + e_t$$

with:

$$Q = \begin{bmatrix} \sigma_\eta^2 & 0 \\ 0 & \sigma_w^2 \end{bmatrix} \; ; \; S = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \; ; \; R = \sigma_\epsilon^2$$

2.2 The steady-state innovations model

A particular case of (1)-(2) is the *steady-state innovations* SS model, see Anderson and Moore (1979), defined by:

$$x_{t+1} = \Phi x_t + \Gamma u_t + E\varepsilon_t \tag{5}$$

$$z_t = Hx_t + Du_t + \varepsilon_t \tag{6}$$

Comparing (5)-(6) with (1)-(2) it is immediate to see that, in this formulation, the errors in the state and observation equations are the same and $C = I$.

The relevance of this case lies in two facts: a) many econometric models in SS have the steady-state innovations structure, see *e.g.*, the SS representation in Example 2.1, and b) when applied to a SS model with this structure, the forecasting, filtering and smoothing algorithms have special convergence properties, which allow the implementation of very efficient and stable computational procedures, see Casals, Sotoca and Jerez (1999) and Casals, Jerez and Sotoca (2000).

2.3 The state-space representation in E⁴

2.3.1 THD format

E⁴ employs the SS representation (1)-(4) for most computations. However, the SS formulation is not adequate for manipulation and storage. Therefore, E⁴ stores the description of a model in a more convenient intermediate format called "THD". Any THD format specification is composed of two matrices: `theta` and `din`, which contain, respectively, the values of the model parameters and a description of its dynamic structure. Besides of `theta` and `din`, a model can be documented by an optional character matrix `lab`, which contains names for the parameters in `theta`. While some analyses may require some editing of `theta` and `lab`, most users will never need to handle the matrix `din`.

Therefore, in E⁴ defining a model consists of a) creating its parameter matrices by means MATLAB commands and b) feeding these matrices to an E⁴ interface function, which generates `theta`, `din` and `lab`.

Example 2.3 (Defining parameter matrices). The following MATLAB commands initialize five matrices, which could be used to define a model in THD format:


```
A=[-.8 NaN;.1 0];
Sigma1=[1 .1; .1 2];
Sigma2=[1  2; .1 2];
Sigma3=[1;2];
Sigma4=[1 NaN;NaN 2];
```

The first command generates the matrix:

$$A = \begin{bmatrix} -.8 & 0 \\ .1 & 0 \end{bmatrix}$$

where the parameter in the position (1,2) is constrained to remain in its present null value because it has been defined using *NaN*. The zero element in the position (2,2) defines a parameter with a null starting value, which will be allowed to change during the estimation process. The second and third commands define different matrices but, when interpreted as covariance matrices by an E⁴ function, they are equivalent because the upper triangle is in fact ignored.

$$\Sigma_1 = \Sigma_2 = \begin{bmatrix} 1 & .1 \\ .1 & 2 \end{bmatrix}$$

The fourth command defines the covariance matrix:

$$\Sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

where the off-diagonal elements are constrained to remain at its present null values. Finally, the last command is a valid MATLAB sentence but, when interpreted as a covariance matrix by an E⁴ function, will generate an error message due to the presence of *NaN*.

2.3.2 Defining SS and unobserved components time series models

The function `ss2thd` translates the SS representation (1)-(4) to THD format. Its syntax is:

```
[theta, din, lab] = ss2thd(Phi, Gam, E, H, D, C, Q, S, R);
```

where the input arguments correspond to the parameter matrices in (1)-(4).

Example 2.4 (Unobserved components time series model). Consider the model from Example 2.2, with $\sigma_\eta^2 = .1$, $\sigma_e^2 = 2$ and $\sigma_w^2 = 1$. To define this model with E⁴ it is first necessary to obtain its SS

representation:

```
Phi=zeros(13,13);
Phi(1,1)=1; Phi(1,2)=1; Phi(2,2)=1; Phi(3,13)=-1;
for i=3:12
   Phi(3,i)=-1;
   Phi(i+1,i)=1;
end
E=zeros(13,2); E(2,1)=1; E(3,2)=1;
H=zeros(1,13); H(1,1)=1; H(1,3)=1;
C=[1]; Q=[.1; 1.0]; S=[0; 0]; R=[2.0];
```

afterwards, the THD form is obtained and displayed using the function `prtmod`:

```
[theta, din, lab] = ss2thd(Phi, [], E, H, [], C, Q, S, R);
prtmod(theta, din, lab);
```

and the output is:

```
************************** Model **************************
Native SS model
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 13
Parameters (* denotes constrained parameter):
PHI(1,1)            1.0000
PHI(2,1)            0.0000
PHI(3,1)            0.0000
PHI(4,1)            0.0000
PHI(5,1)            0.0000
PHI(6,1)            0.0000
PHI(7,1)            0.0000
PHI(8,1)            0.0000
PHI(9,1)            0.0000
PHI(10,1)           0.0000
PHI(11,1)           0.0000
PHI(12,1)           0.0000
PHI(13,1)           0.0000

... [part of this listing has been deliberately omitted]

Q(1,1)              0.1000
Q(2,2)              1.0000
S(1,1)              0.0000
S(2,1)              0.0000
R(1,1)              2.0000
```

If the model is to be estimated, all the parameters except the variances should keep its present values. These fixed-value constraints can be imposed with the additional commands:

```
theta =[theta ones(214,1)]; theta(210,2)= 0; theta(211,2)= 0;
theta(214,2)= 0;
prtmod(theta, din, lab);
```

and the resulting output is identical to previous output, where the constrained parameters are marked with an asterisk.

3. Simple models

It would be arduous if we had to define manually the matrices in (1)-(4) for every model. Therefore, E⁴ provides functions to create these matrices for frequently used models. The following table describes the models specifically supported, as well as the combinations of options available:

Support for: Simple models	Missing data	Stationarity/ Nonstationarity	GARCH errors
VARMAX	YES	YES	YES
Structural econometric	YES	YES	NO
Single-output transfer functions	YES	YES	YES
State-Space (including unobserved components time series models)	YES	YES	YES†

† Only for models in steady-state innovations form

This Section describes both, the formulation of these models and the functions supporting them. Besides "simple models", the Toolbox supports a new concept called "composite models". Composite models are given by a combination of two or more simple models into a more complex formulation, allowing unusual features such as multiple seasonal patterns or observation errors. This option is described in Section 4.

3.1 Structural econometric models

A structural econometric model can be formulated as:

$$FR(B)\,FS(B^S)\,y_t = G(B)u_t + AR(B)\,AS(B^S)\,\varepsilon_t \qquad (7)$$

where S denotes the length of the seasonal period, B is the backshift operator, such that for any sequence x_t: $B^{\pm k}x_t = x_{t\mp k}$, and y_t is a $(m\times 1)$ vector of endogenous variables, u_t is a $(r\times 1)$ vector of exogenous variables, ε_t is a $(m\times 1)$ vector of white noise errors and

$$FR(B) = FR_0 + FR_1B + \ldots + FR_pB^p \;;\; FS(B^S) = FS_0 + FS_1B^S + \ldots + FS_PB^{S\cdot P}$$

$$G(B) = G_0 + G_1B + \ldots + G_gB^g \;;\; AR(B) = AR_0 + AR_1B + \ldots + AR_qB^q$$

$$AS(B) = AS_0 + AS_1B^S + \ldots + AS_QB^{S\cdot Q}$$

The characteristic feature of this formulation consists of allowing for a contemporary relationship

between the endogenous variables, given by the matrices FR_0 and FS_0. This formulation includes, as particular cases, the linear regression model and the simultaneous equations model.

E⁴ implementation. The function `str2thd` obtain the THD specification for structural econometric models:

```
[theta,din,lab] = str2thd({[FR0 ... FRp],[FS0 ... FSps], ...
                           [AR0 ... ARq],[AS0 ... ASqs],v,s,[G0 ... Gg],r)
```

where `v` is the error covariance matrix and `r` is the number of exogenous variables.

Example 3.1 (Defining structural econometric models). Consider the model:

$$\left[\begin{pmatrix} 1 & -.3 \\ -.7 & 1 \end{pmatrix} + \begin{pmatrix} -.4 & 0 \\ .5 & 0 \end{pmatrix} B\right]\begin{bmatrix} y_{1t} \\ y_{2t} \end{bmatrix} = \begin{pmatrix} .3 & 0 \\ 0 & .5 \end{pmatrix}\begin{bmatrix} u_{1t} \\ u_{2t} \end{bmatrix} + \begin{bmatrix} \varepsilon_{1t} \\ \varepsilon_{2t} \end{bmatrix} \qquad V\begin{bmatrix} \varepsilon_{1t} \\ \varepsilon_{2t} \end{bmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & .9 \end{pmatrix}$$

The first step to obtain its definition in THD format consists of defining the input arguments to `str2thd`. This can be done with the following MATLAB commands:

```
FR0=[ 1  -.3; -.7  1 ]; FR1=[-.4 NaN;  .5 NaN];
G0 =[ .3 NaN; NaN  .5]; v  =[1.0  .9];
```

and afterwards, the THD form is obtained and displayed with:

```
[theta, din, lab] = str2thd{[FR0 FR1],[],[],[],v,1,[G0],2);
prtmod(theta, din, lab);
```

and the output is:

```
************************** Model ****************************
Structural model (innovations model)
2 endogenous v., 2 exogenous v.
Seasonality: 1
SS vector dimension: 2
Parameters (* denotes constrained parameter):
FR0(1,1)          1.0000
FR0(2,1)         -0.7000
FR0(1,2)         -0.3000
FR0(2,2)          1.0000
FR1(1,1)         -0.4000
FR1(2,1)          0.5000
G0(1,1)           0.3000
G0(2,2)           0.5000
V(1,1)            1.0000
V(2,2)            0.9000
*************************************************************
```

E⁴ includes a function to obtain the SS model matrices from the THD representation of any model. Its general syntax is:


```
[Phi, Gam, E, H, D, C, Q, S, R] = thd2ss(theta, din);
```

and the output corresponding to this example is:

```
Phi =
    0.3165         0
   -0.2785         0

Gam =
    0.1202    0.0601
   -0.1058   -0.0529

E =
    0.4006    0.1202
   -0.3525   -0.1058

H =
     1     0
     0     1
D =
    0.3797    0.1899
    0.2658    0.6329

C =
    1.2658    0.3797
    0.8861    1.2658

Q =
    1.0000         0
         0    0.9000

S =
    1.0000         0
         0    0.9000
R =
    1.0000         0
         0    0.9000
```

3.2 VARMAX models

The VARMAX model is defined by:

$$FR(B)\,FS(B^S)\,y_t = G(B)u_t + AR(B)\,AS(B^S)\,\varepsilon_t \tag{8}$$

where y_t, u_t, ε_t are defined in (7) and:

$$FR(B) = I + FR_1 B + \ldots + FR_p B^p;\ FS(B^S) = I + FS_1 B^S + \ldots + FS_P B^{S\cdot P}$$
$$G(B) = G_0 + G_1 B + \ldots + G_g B^g\ ;\ AR(B) = I + AR_1 B + \ldots + AR_q B^q$$
$$AS(B) = I + AS_1 B^S + \ldots + AS_Q B^{S\cdot Q}$$

E⁴ implementation. The function `arma2thd` obtains the THD specification for VARMAX models:

```
[theta,din,lab] = arma2thd([FR1 ... FRp],[FS1 ... FSps], ...
                           [AR1 ... ARq],[AS1 ... ASqs],v,s,[G0 ... Gg],r)
```

The input arguments are analogous to that of `str2thd`.

Example 3.2 (Defining VARMAX models): The model:

$$\left[\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}+\begin{pmatrix}-.8 & 0\\ -.5 & -.7\end{pmatrix}B+\begin{pmatrix}-.45 & 0\\ 0 & 0\end{pmatrix}B^2\right]\left[\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}+\begin{pmatrix}-.2 & 0\\ 0 & -.3\end{pmatrix}B^4\right]\boldsymbol{y}_t=$$
$$\left[\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}+\begin{pmatrix}-.4 & 0\\ 0 & -.3\end{pmatrix}B\right]\boldsymbol{a}_t \qquad V(\boldsymbol{a}_t)=\begin{bmatrix}.9 & 0\\ 0 & .9\end{bmatrix}$$

can be defined in THD format and displayed with the following MATLAB commands:

```
FR1 =[-.8  NaN; -.5 -.7];
FR2 =[-.45 NaN; NaN NaN];
FS1 =[-.2  NaN; NaN -.3];
AR1 =[-.4  NaN; NaN -.3];
v   =[.9   .9];
[theta, din, lab] = arma2thd([FR1 FR2],[FS1],[AR1],[],v,4);
prtmod(theta, din, lab);
```

Note that an empty matrix, `[]`, should appear where the model does not include the corresponding structure. In this case there was no seasonal moving average factor. Finally, the output displayed by `prtmod` is the following:

```
************************** Model ****************************
VARMAX model (innovations model)
2 endogenous v., 0 exogenous v.
Seasonality: 4
SS vector dimension: 12
Parameters (* denotes constrained parameter):
FR1(1,1)          -0.8000
FR1(2,1)          -0.5000
FR1(2,2)          -0.7000
FR2(1,1)          -0.4500
FS1(1,1)          -0.2000
FS1(2,2)          -0.3000
AR1(1,1)          -0.4000
AR1(2,2)          -0.3000
V(1,1)             0.9000
V(2,2)             0.9000
*************************************************************
```

3.3 Transfer function models

The third simple specification is the single-output transfer function model, which can be formulated as:

$$y_t = \frac{\omega_1(B)}{\delta_1(B)}u_{1t} + \ldots + \frac{\omega_r(B)}{\delta_r(B)}u_{rt} + \frac{\theta(B)\,\Theta(B^S)}{\phi(B)\,\Phi(B^S)}\varepsilon_t \qquad (9)$$

where y_t is the value of the endogenous variable at time t, $\boldsymbol{u}_t=[u_{1t},\ldots,u_{rt}]^T$ is a $(r\times 1)$ vector of

exogenous variables, ε_t is a white noise error and

$$\omega_i(B) = \omega_{i0}+\omega_{i1}B+\omega_{i2}B^2+\ldots+\omega_{in_i}B^{n_i}\ ;\ i=1,2,\ldots,r$$
$$\delta_i(B) = 1+\delta_{i1}B+\ldots+\delta_{ind_i}B^{nd_i}\ ;\ i=1,2,\ldots,r$$
$$\phi(B) = 1+\phi_1B+\ldots+\phi_pB^p\ ;\ \Phi(B^S) = 1+\Phi_1B^S+\ldots+\Phi_PB^{P\cdot S}$$
$$\theta(B) = 1+\theta_1B+\ldots+\theta_qB^q\ ;\ \Theta(B^S) = 1+\Theta_1B^S+\ldots+\Theta_QB^{Q\cdot S}$$

E⁴ implementation. The function `tf2thd` obtains the THD format for transfer functions:

```
[theta,din,lab] = tf2thd([fr1 ... frp], [fs1 ... fsps], ...
        [ar1 ... arq],[as1 ... asqs],v,s,[w1; ...; wr),[d1; ...; dr]);
```

Example 3.3 (Defining transfer functions). Given the transfer function:

$$y_t = \frac{.3+.6B}{1-.5B}u_{1t} + (.3B+.4B^2)u_{2t} + \frac{.3B}{1-.1B-.2B^2}u_{3t} + \frac{1-.8B}{1-.6B}\varepsilon_t\ ;\ \sigma_e^2=1$$

its definition in THD format is obtained as follows:

```
w1 = [ .3 .6 NaN]; d1 = [-.5 NaN];
w2 = [NaN .3 .4 ]; d2 = [NaN NaN];
w3 = [NaN .3 NaN]; d3 = [-.1 -.2];
fr = [-.6]; ar = [-.8];
v  = [1.0];
[theta,din,lab]=tf2thd(fr,[],ar,[],v,1,[w1;w2;w3],[d1;d2;d3]);
prtmod(theta,din,lab);
```

and the corresponding `prtmod` output is:

```
************************** Model ****************************
Transfer function model (innovations model)
1 endogenous v., 3 exogenous v.
Seasonality: 1
SS vector dimension: 5
Parameters (* denotes constrained parameter):
FR(1,1)           -0.6000
AR(1,1)           -0.8000
W1(1,1)            0.3000
W1(2,1)            0.6000
W2(2,1)            0.3000
W2(3,1)            0.4000
W3(2,1)            0.3000
D1(1,1)           -0.5000
D3(1,1)           -0.1000
D3(2,1)           -0.2000
V(1,1)             1.0000
*************************************************************
```

4. Composite models

A model that combines the formulation of several models is called a *composite model*. E⁴ allows for two types of model composition: *nested models* and *component models*.

First, a model is said to be nested if it is constructed by a combination of multiplicative factors. For example, the famous airline model can be viewed as the result of nesting a regular IMA(1,1) process with a seasonal IMA(1,1) process.

On the other hand, a component model is obtained by the addition of several dynamic structures. For example, one could define the model for a trend-cycle decomposition by adding an ARIMA model for the trend and an ARMA model for the cycle.

The following sub-sections provide details and examples about the implementation of composite models in E⁴.

4.1 Nested models

E⁴, allows for two types of model nesting: nesting in inputs and nesting in noise.

4.1.1 Nesting in inputs

A model is said to be nested in inputs if some exogenous variable is substituted by a model describing its dynamic and stochastic structure. After doing so, the exogenous variables become endogenous and, therefore, we can also describe this operation as "endogeneization". The following example illustrates in a general SS framework:

Example 4.1 (Endogeneization). Following Terceiro (1990), the SS approach to endogeneization proceeds as follows. Assume that the model for the vector y_t is:

$$x_{t+1}^a = \Phi^a x_t^a + \Gamma^a u_t + E^a w_t^a \tag{10}$$

$$y_t = H^a x_t^a + D^a u_t + C^a v_t^a \tag{11}$$

whereas the exogenous variables, u_t, follow the model:

$$x_{t+1}^b = \Phi^b x_t^b + E^b w_t^b \tag{12}$$

$$u_t = H^b x_t^b + C^b v_t^b \tag{13}$$

being the errors in (10)-(11) and (12)-(13) mutually independent. Substituting (13) in (10)-(11) yields:

$$x_{t+1}^a = \Phi^a x_t^a + \Gamma^a (H^b x_t^b + C^b v_t^b) + E^a w_t^a \tag{14}$$

$$y_t = H^a x_t^a + D^a (H^b x_t^b + C^b v_t^b) + C^a v_t^a \tag{15}$$

Eqs. (14) and (12) can be easily combined in a single state equation:

$$\begin{bmatrix} x_{t+1}^a \\ x_{t+1}^b \end{bmatrix} = \begin{bmatrix} \Phi^a & \Gamma^a H^b \\ 0 & \Phi^b \end{bmatrix} \begin{bmatrix} x_t^a \\ x_t^b \end{bmatrix} + \begin{bmatrix} E^a & \Gamma^a C^b & 0 \\ 0 & 0 & E^b \end{bmatrix} \begin{bmatrix} w_t^a \\ v_t^b \\ w_t^b \end{bmatrix} \tag{16}$$

and Eqs. (15) and (13) can be combined in a single observation equation:

$$\begin{bmatrix} y_t \\ u_t \end{bmatrix} = \begin{bmatrix} H^a & D^a H^b \\ 0 & H^b \end{bmatrix} \begin{bmatrix} x_t^a \\ x_t^b \end{bmatrix} + \begin{bmatrix} C^a & D^a C^b \\ 0 & C^b \end{bmatrix} \begin{bmatrix} v_t^a \\ v_t^b \end{bmatrix} \tag{17}$$

E⁴ implementation. To obtain in E⁴ the THD formulation of a model nested in inputs, one should follow a three step procedure:

Step 1) Obtain the THD formulation of the models for the endogenous and exogenous variables, using functions as `arma2thd` or `tf2thd`.

Step 2) Feed the THD forms obtained to the function `stackthd`, which arranges the individual THD formats into a single ("stacked") THD description. The syntax is:

```
[theta, din, lab] = stackthd(t1, d1, t2, d2, l1, l2);
```

where the input arguments `t1, d1, l1` and `t2, d2, l2` are the THD forms obtained in Step 1). If necessary, the output arguments can be feed again to `stackthd`, to continue recursively the stacking process.

Step 3) Translate the stacked model to the final nested formulation using the function `nest2thd`, which syntax is:

```
[theta, din, lab] = nest2thd(theta, din, nestwhat, lab);
```

where the input arguments `theta`, `din` and `lab` were the final results of Step 2), and `nestwhat` is a binary flag to choose between nesting in inputs (if `nestwhat` is equal to one) or in errors (if `nestwhat` is equal to zero).

Example 4.2 (Endogeneization in a transfer function): Given the transfer function:

$$y_t = \frac{.3 + .6B}{1 - .5B} u_{1t} + \frac{1 - .8B}{1 - .6B} \varepsilon_t \; ; \; \sigma_\varepsilon^2 = 1$$

where u_{1t} is such that $(1 - .7B)u_{1t} = a_t \; ; \; \sigma_a^2 = .3$. In a standard analytic framework, obtaining forecasts for y_t requires first to forecast the exogenous variable, u_{1t}, and afterwards feed these forecasts to the model. An endogeneized model is an effective way to: a) perform both steps as a single operation and b) taking into account the uncertainty affecting the forecasts for the input, which are often ignored by standard software. The code required to follow steps 1) to 3) is:

Step 1) Obtain the THD representation for both models:

```
w1 = [ .3 .6]; d1 = [-.5];
fr = [-.6];    ar = [-.8];
v  = [1.0];
[t1,d1,l1]=tf2thd(fr,[],ar,[],v,1,[w1],[d1]);
[t2,d2,l2]=arma2thd(-.7,[],[],[],.3,1);
```

Step 2) Combine the model for the endogenous variable with the model for the input into a single "stacked" THD representation:

```
[theta, din, lab] = stackthd(t1, d1, t2, d2, l1, l2);
```

Step 3) Translate the stacked model to the final nested formulation and display a description of its structure:

```
[theta, din, lab] = nest2thd(theta, din, 1, lab);
prtmod(theta,din,lab);
```

The output from `prtmod` is:

```
*************************** Model ***************************
Nested model in inputs (innovations model)
2 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 3
Submodels:
{
  Transfer function model (innovations model)
  1 endogenous v., 1 exogenous v.
  Seasonality: 1
  SS vector dimension: 2
  Parameters (* denotes constrained parameter):
  FR(1,1)            -0.6000
  AR(1,1)            -0.8000
  W1(1,1)             0.3000
  W1(2,1)             0.6000
  D1(1,1)            -0.5000
  V(1,1)              1.0000
  ---------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)           -0.7000
  V(1,1)              0.3000
  ---------------
}
*************************************************************
```

Note that the resulting model has two endogenous variables, y_t and u_{1t}, and no exogenous variable.

4.1.2 Nesting in errors

Nesting in errors consists of defining the noise structure of an econometric model as a multiplicative combination of several dynamic factors. The only requisite for these factors is that their SS equivalent representation should be an innovations model, see (5)-(6). This is not a severe restriction, as most simple models supported by E[4] satisfy it. Two relevant applications of this type of nesting consist of: a) separating unit roots from stationary or invertible factors and b) representing a time series with multiple seasonal cycles.

Example 4.3 (Unit roots). Assume the following ARIMA(1,1,0) model:

$$(1 - .7B)\nabla z_t = a_t \; ; \; \sigma_a^2 = .2$$

where $\nabla \equiv 1 - B$. The standard procedure to deal with non-stationary processes like this consists of eliminating the unit root by differencing the time series. For some applications - *e.g.*, forecasting the level of the time series, z_t, or interpolating missing values - it is more convenient to work directly with the factored AR(2) model:

$$(1 - .7B)(1 - B)z_t = a_t$$

The THD format of this model is obtained and displayed with the following code:

```
[t1, d1, l1] = arma2thd([-1], [], [], [], 1, 1);
[t2, d2, l2] = arma2thd([-.7], [], [], [], .2, 1);
[ts, ds, ls] = stackthd(t1, d1, t2, d2, l1, l2);
[tn, dn, ln] = nest2thd(ts, ds, 0, ls);
prtmod(tn, dn, ln);
```

and the corresponding `prtmod` output is:

```
************************** Model **************************
Nested model in errors (innovations model)
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 2
Submodels:
{
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -1.0000
  --------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -0.7000
  V(1,1)             0.2000
  --------------
}
************************************************************
```

where the only error variance relevant for `nest2thd` is that of the nested model.

Example 4.4 (Multiple seasonal factors). Consider a time series observed once each ten days, z_t, with the following structure:

$$\nabla_{36}\nabla_9\nabla_3\nabla z_t = (1 - .6B^{36})(1 - .7B^9)(1 - .8B^3)(1 - .9B)a_t \quad , \quad \sigma_a^2 = .2$$

where $\nabla_{36} \equiv (1 - B^{36}), \nabla_9 = (1 - B^9)$ and $\nabla_3 = (1 - B^3)$.

The THD representation of this model can be obtained with:

```
[t1, d1, l1] = arma2thd([], [-1], [], [-.6], 1, 36);
[t2, d2, l2] = arma2thd([], [-1], [], [-.7], 1, 9);
[t3, d3, l3] = arma2thd([-1], [-1], [-.9], [-.8], .2, 3);
[ts1, ds1, ls1] = stackthd(t1, d1, t2, d2, l1, l2);
[ts2, ds2, ls2] = stackthd(ts1, ds1, t3, d3, ls1, l3);
[tn, dn, ln] = nest2thd(ts2, ds2, 0, ls2);
prtmod(tn, dn, ln);
```

and the corresponding output is:

```
************************** Model **************************
Nested model in errors (innovations model)
1 endogenous v., 0 exogenous v.
Seasonality: 36
SS vector dimension: 49
Submodels:
{
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 36
  SS vector dimension: 36
  Parameters (* denotes constrained parameter):
  FS1(1,1)          -1.0000
  AS1(1,1)          -0.6000
  --------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
```



```
  Seasonality: 9
  SS vector dimension: 9
  Parameters (* denotes constrained parameter):
  FS1(1,1)          -1.0000
  AS1(1,1)          -0.7000
  --------------
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 3
  SS vector dimension: 4
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -1.0000
  FS1(1,1)          -1.0000
  AR1(1,1)          -0.9000
  AS1(1,1)          -0.8000
  V(1,1)             0.2000
  --------------
}
************************************************************
```

4.2 Component models

A component model is defined as the sum of several components, having all of them a simple or composite model describing its particular features. Two important cases of component models are Structural Time Series Models (STSM), see Harvey (1989) and models with observation errors, see Terceiro (1990).

E⁴ implementation. A component model is defined in E⁴ following a three step procedure, very similar to that described in Section 4.1 for nested models. In fact, Steps 1) and 2) are identical. Step 3) is similar, replacing the call to `nest2thd` for a similar call to `comp2thd`. The syntax of this function is:

```
[theta, din, lab] = comp2thd(ts, ds, ls);
```

where the input arguments `ts`, `ds` and `ls` are the stacked THD format of the models to be composed.

Example 4.5. (The HP filter model). As it is well known, the famous Hodrick and Prescott (1980) filter implicitly assumes that the series follows the unobserved components model:

$$y_t = T_t + e_t$$

$$T_{t+1} = T_t + \Delta_t$$

$$\Delta_{t+1} = \Delta_t + \eta_t$$

which is again a particular case of the "stochastic trend model", see Harvey (1989) and Example 2.2. This model can also be written as an integrated random-walk with a white noise observation error:

$$y_t = T_t + e_t$$

$$(1 - B)T_{t+1} = \eta_t$$

and η_t and e_t are often assumed to be independent errors, with a noise-to variance ratio of 1/1600 (if y_t is a quarterly time series) so, if the variance of e_t is one, the error covariance matrix would be:

$$\text{COV}\begin{bmatrix}\eta_t \\ e_t\end{bmatrix} = \begin{bmatrix}1/1600 & 0 \\ 0 & 1\end{bmatrix}$$

The corresponding THD format can be obtained with the code:

```
e4init
% Defines the implicit model for the HP quarterly filter as an
%   integrated random walk with a white noise observation error
[t1, d1, l1] = arma2thd([-2 1], [], [], [], 1/1600, 1);
[t2, d2, l2] = arma2thd([], [], [], [], 1, 1);
[ts, ds, ls] = stackthd(t1, d1, t2, d2, l1, l2);
[theta, din, lab] = comp2thd(ts, ds, ls);
prtmod(theta, din, lab);
```

which yields the following output:

```
*************************** Model ****************************
Components model
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 2
Submodels:
{
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 2
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -2.0000
  FR2(1,1)           1.0000
  V(1,1)             0.0006
  ---------------
  White noise model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 0
  Parameters (* denotes constrained parameter):
  V(1,1)             1.0000
  ---------------
}
*************************************************************
```

5. Models with GARCH errors

Most econometric models assume that errors are homoscedastic. To generalize this assumption, Engle (1982) introduced a new class of stochastic processes with time-varying conditional variances. For a comprehensive survey, see Bollerslev *et al.* (1994).

E⁴ allows one to combine any VARMAX model, transfer function or SS in the steady-state

innovations form (5)-(6), with a vector GARCH process for the error $\boldsymbol{\varepsilon}_t$. This process is defined by the unconditional moments $\mathrm{E}[\boldsymbol{\varepsilon}_t] = \mathbf{0}$, $V[\boldsymbol{\varepsilon}_t] = \boldsymbol{\Sigma}_\varepsilon$; and the conditional moments: $\mathrm{E}_{t-1}[\boldsymbol{\varepsilon}_t] = \mathbf{0}$, $\mathrm{E}_{t-1}[\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T] = \boldsymbol{\Sigma}_t$, where $\mathrm{E}_{t-1}[\,]$ denotes the expectation of the argument conditional to the information available in t-1. In a GARCH model, the conditional variances, $\boldsymbol{\Sigma}_t$, are such that:

$$[I - \mathrm{B}(B)]\,\mathrm{vech}(\boldsymbol{\Sigma}_t) = \mathrm{vech}(w) + \mathrm{A}(B)\,\mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T) \tag{18}$$

where vech( ) stands for the vector-half operator, which stacks the lower triangle of a $N\times N$ matrix as a $[N(N+1)/2]\times 1$ vector; and the polynomial matrices are given by:

$$\mathrm{B}(B) = \sum_{i=1}^{s_e} \mathrm{B}_i B^i \quad ; \quad A(B) = \sum_{i=1}^{p_e} A_i B^i$$

E⁴ manages model (18) in its alternative VARMAX representation. To derive it, consider the process $\boldsymbol{\nu}_t = \mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T) - \mathrm{vech}(\boldsymbol{\Sigma}_t)$, such that $\mathrm{E}_{t-1}[\boldsymbol{\nu}_t] = \mathbf{0}$. Then $\mathrm{vech}(\boldsymbol{\Sigma}_t) = \mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T) - \boldsymbol{\nu}_t$. Substituting this expression in (18) and rearranging some terms yields:

$$[I - \mathrm{A}(B) - \mathrm{B}(B)]\,\mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T) = \mathrm{vech}(w) + [I - \mathrm{B}(B)]\boldsymbol{\nu}_t \tag{19}$$

Eq. (19) defines a VARMAX model for $\mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T)$, which can be written in compact form as:

$$\mathrm{vech}(\boldsymbol{\varepsilon}_t\boldsymbol{\varepsilon}_t^T) = \mathrm{vech}(\boldsymbol{\Sigma}_\varepsilon) + \boldsymbol{N}_t \tag{20}$$

$$[\mathrm{I} + \bar{\mathrm{A}}(B)]\boldsymbol{N}_t = [\mathrm{I} + \bar{\mathrm{B}}(B)]\boldsymbol{\nu}_t \tag{21}$$

where $\bar{\mathrm{A}}(B) = -[\mathrm{A}(B) + \mathrm{B}(B)]$ and $\bar{\mathrm{B}}(B) = -\mathrm{B}(B)$. This is the formulation supported in E⁴. Note that: 1) if the VAR polynomial in (21) has roots on the unit radius circle, then the process has some IGARCH (*Integrated-GARCH*) components and 2) the formulation (20)-(21) does not assure the eigenvalues of $\boldsymbol{\Sigma}_t$ to be non-negative for all t.

E⁴ implementation Formulation of models with conditional heteroscedastic errors in THD format is similar to that of composite models. First, it is necessary to obtain the THD formulation of: a) a VARMAX or transfer function model for the mean and b) a VARMAX model equivalent to the ARCH, GARCH or IGARCH structure desired. The full model can then be defined using the `garc2thd` function, which has the following syntax:

```
[theta, din, lab] = garc2thd(t1, d1, t2, d2, lab1, lab2);
```

where `t1-d1` is the THD format associated with the model for the mean, `t2-d2` is the THD format associated with the VARMAX model for the variance, and `lab1` and `lab2` are optional parameters with labels for the parameters in `t1` and `t2`, respectively.

Example 5.1 (Defining models with GARCH errors). Consider the following ARMA(2,1) model with GARCH(1,1) errors, in conventional notation:

$$y_t = \frac{1 - .8B}{1 - .7B + .3B^2}\varepsilon_t \; ; \; \varepsilon_t \sim iid(0, .01) \; ; \; \varepsilon_t | \, \Omega_t \sim iid(0, h_t^2) \; ;$$

$$h_t^2 = .002 + .1\varepsilon_{t-1}^2 + .7h_{t-1}^2$$

which, in the ARMA representation supported by E^4 becomes:

$$y_t = \frac{1 - .8B}{1 - .7B + .3B^2}\varepsilon_t \text{, such that: } \varepsilon_t^2 = .01 + N_t, \; (1 - .8B)N_t = (1 - .7B)v_t$$

The following code defines and displays the structure of this model:

```
% Model for the mean
[t1, d1, lab1] = arma2thd([-.7 .3], [], [-.8], [], [.01], 1);
% Model for the conditional variance
[t2, d2, lab2] = arma2thd([-.8], [], [-.7], [], [.01], 1);
% Full model
[theta, din, lab] = garc2thd(t1, d1, t2, d2, lab1, lab2);
prtmod(theta, din, lab);
```

generating the output:

```
************************** Model ****************************
GARCH model (innovations model)
1 endogenous v., 0 exogenous v.
Seasonality: 1
SS vector dimension: 2
Endogenous variables model:
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 2
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -0.7000
  FR2(1,1)           0.3000
  AR1(1,1)          -0.8000
  V(1,1)             0.0100
  ---------------
GARCH model of noise:
  VARMAX model (innovations model)
  1 endogenous v., 0 exogenous v.
  Seasonality: 1
  SS vector dimension: 1
  Parameters (* denotes constrained parameter):
  FR1(1,1)          -0.8000
  AR1(1,1)          -0.7000
  ---------------
*************************************************************
```


6. Algorithms

This Section documents the basic functions used in E^4 for estimation, filtering, smoothing and simulation.

6.1. Evaluation of the likelihood function

There are many ways to compute the likelihood of (1)-(2). De Jong (1988) derives the expression:

$$\begin{aligned}\ell(Z/U,\theta) \; = \; & \log|P_1| \; + \bar{x}_1^T P_1^{-1} \bar{x}_1 \; + \sum_{t=1}^{N} \log|B_t| \; + \sum_{t=1}^{N} \tilde{z}_t^T B_t^{-1} \tilde{z}_t \; + \\ & + \log|P_1^{-1} + W_N| \; - \; (P_1^{-1}\bar{x}_1 + w_N)^T (P_1^{-1} + W_N)^{-1} (P_1^{-1}\bar{x}_1 + w_N)\end{aligned} \tag{22}$$

where $Z = [z_1, z_2, ..., z_N]$, $U = [u_1, u_2, ..., u_N]$ and θ is a vector that contains the unknown values in Φ, Γ, E, H, C, D, Q, R and S. The innovations $\tilde{z}_t$ and their conditional covariance matrices B_t in (22) result from a Kalman filter initialized with $\bar{x}_1 = 0$ and $P_1 = 0$, from now KF(**0**, **0**):

$$\hat{x}_{t+1/t} = \Phi \hat{x}_{t/t-1} + \Gamma u_t + K_t \tilde{z}_t \tag{23}$$

$$\tilde{z}_t = z_t - H\hat{x}_{t/t-1} - Du_t \tag{24}$$

$$K_t = (\Phi P_{t/t-1} H^T + ESC^T) B_t^{-1} \tag{25}$$

$$P_{t+1/t} = \Phi P_{t/t-1} \Phi^T + EQE^T - K_t B_t K_t^T \tag{26}$$

$$B_t = HP_{t/t-1} H^T + CRC^T \tag{27}$$

where $\hat{x}_{t+1/t}$ and $\hat{x}_{t/t-1}$ are the one-step-ahead conditional expectations of x_{t+1} and x_t, $P_{t+1/t}$ and $P_{t/t-1}$ are the corresponding conditional covariances and K_t is the Kalman filter gain. The variables w_N and W_N in (22) result from the recursions:

$$w_t = w_{t-1} + \bar{\Phi}_{t-1}^T H^T B_t^{-1} \tilde{z}_t \; ; \; w_0 = 0 \tag{28}$$

$$W_t = W_{t-1} + \bar{\Phi}_{t-1}^T H^T B_t^{-1} H \bar{\Phi}_{t-1} \; ; \; W_0 = 0 \tag{29}$$

$$\bar{\Phi}_t = (\Phi - K_t H)\bar{\Phi}_{t-1} \; ; \; \bar{\Phi}_0 = I \tag{30}$$

The terms $\log|P_1|$, $\log|P_1^{-1}+W_N|$ and $(P_1^{-1}\bar{x}_1+w_N)^T(P_1^{-1}+W_N)^{-1}(P_1^{-1}\bar{x}_1+w_N)$ in (22) depend on the initial conditions of the Kalman filter. When the system is stationary, all of them can be computed. If there are some unit roots, the first term can be approximated taking into account only its stationary part, see Casals and Sotoca (1997). The other expressions are not problematic because P_1^{-1} is finite, see De Jong and Chu-Chun-Lin (1994).

E⁴ implementation. Starting from a model in THD format and a sample, the functions `lfmod` and `lffast` compute the value of the gaussian log-likelihood function for any time- invariant model with homoscedastic errors. Their syntax is:

```
[f, innov, ssvect] = lfmod(theta, din, z);
[f, innov, ssvect] = lffast(theta, din, z);
```

The function `lfmod` computes the log-likelihood of a standard sample, see Terceiro (1990, Chapter 4) for details. The function `lffast` is a faster version of `lfmod` because it takes advantage of the innovations structure of many econometric models; see Casals, Sotoca and Jerez (1999).

The input arguments of these functions are a THD format specification, `theta`, `din` and the data matrix `z`. Each of these functions compute the value of the following output arguments: 1) `f`, a scalar that contains the value of the log-likelihood function in `theta`, see Eq. (22), 2) `innov`, a matrix of one-step-ahead forecast errors, see Eq. (24) and 3) `ssvect`, a matrix of estimates of the state variables. Its t-th row contains the filtered estimate of the state vector at time t, conditional on the information available up to t-1, see Eq. (23).

The gaussian log-likelihood of models with GARCH errors, see Section 5, is computed by the function `lfgarch`, which synopsis is:

```
[f, innov, hominnov, ssvect] = lfgarch(theta, din, z);
```

where all the arguments are the same as those of `lfmod` except `hominnov`, which is a matrix of residuals standardized with the square root of the conditional variances.

6.2. Initial conditions

The values $\bar{x}_1$ and P_1 in (22) should be computed taking into account not only the stationarity of the series, but also the stochastic structure of u_t, see Casals and Sotoca (1997, 2000). In E⁴ adequate initial conditions are chosen automatically. The default options can be manually overridden using the function `sete4opt`.

6.3. Missing data

The approach in E⁴ to compute the likelihood of a sample with missing data consists of assuming that, when a component of z_t is missing, the corresponding variance in B_t is (close to) infinity. This implies that the corresponding value in the Kalman filter gain is zero. To see this, consider Eq. (23)-(27) and assume that, for some t, B_t is arbitrarily close to infinite and, therefore, B_t^{-1} is zero. In this situation, the filter propagation collapses to:

$$\hat{x}_{t+1/t} = \Phi\hat{x}_{t/t-1} + \Gamma u_t \tag{31}$$

$$P_{t+1/t} = \Phi P_{t/t-1}\Phi^T + EQE^T \tag{32}$$

E⁴ implementation. The function `lfmiss` compute the value of the log-likelihood of a sample with missing data. Its syntax is the same as that `lfmod`. The data matrix feed as input argument should contain the MATLAB special value *NaN* (not-a-number) in the positions with missing values.

6.4. Computation of the gradient and the information matrix

The exact gradient and information matrix of the log-likelihood function are relevant to both, the iterative estimation process and many inference procedures, see Engle (1984). Their exact expressions for a general SS model were given by Terceiro (1990).

E⁴ implementation. The general syntax for the functions dealing with gradient computation is:

```
g = gmod(theta, din, z);
```

where `gmod` computes the derivatives of `lfmod` and `lffast`. Other gradient-related functions are `gmiss`, for samples with missing data, and `ggarch`, which computes the derivatives of `lfgarch`. Details about the gradient of the log-likelihood are given in Terceiro (1990, Appendix B). As in the case of the gradient, there are three functions dealing with computation of the information matrix: `imod`, `imiss`, and `igarch`. The input and output arguments for these functions are the same. For example, the syntax of `imod` is:

```
[std, corrm, varm, Im] = imod(theta, din, z, aprox);
```

The function `imod` computes the information matrix of `lfmod` and `lffast`, `imiss` computes the information matrix of `lfmiss` and `igarch` does the same for `lfgarch`. The input argument `approx` indicates whether the calculations should be exact or approximate, being the latter option computationally more efficient, see Watson and Engle (1983). The information matrix for models with GARCH errors is always an approximation. If the model may be misspecified or its errors are non-

normal, the function `imodg` computes the quasi-maximum likelihood information matrix of `lfmod` and `lffast`, according to the proposal of White (1982).

The output arguments are `std`, the standard deviation of the values in `theta`, `corrm`, the correlation matrix between these parameters, `varm`, which is the covariance matrix and `Im`, which is the exact information matrix in the case of `imod` and `imiss`, see Terceiro (1990, Appendices C and D) and Terceiro (1999).

6.5. Forecasting

Out-of-sample predictions, together with their mean square errors, can be generated by the Kalman filter by extending the data with a set of missing data. A sequence of missing data at the end of the sample will therefore produce a set of multi-step forecasts.

E^4 implementation. Forecasting with E^4 requires to obtain the THD representation of the model, estimate its parameters and then call the function `foremod`. This function has the following syntax:

```
[yf, Bf] = foremod(theta, din, z, k, u);
```

where `z` is a data matrix containing the values of the endogenous and exogenous variables, `k` is the forecast horizon and `u` contains the data of the exogenous variables for the forecast horizon. The output arguments are forecasts of the endogenous variables (`yf`) and its corresponding covariances (`Bf`). The function `foremiss` allows for missing values in the sample. Forecasts for model with GARCH errors are computed by `foregarc`. Its syntax is:

```
[yf, Bf, vf] = foregarc(theta, din, z, k, u);
```

where the output argument `vf` contains forecasts of the conditional covariance.

6.6. Simulation

The functions `simmod` and `simgarch` generate a random sample from any model in THD format. The general syntax of these functions is:

```
y = simmod(theta, din, N, u);
```

where the arguments are a model in THD format, the number of observations to be generated (`N`) and the exogenous variable data matrix (`u`). Both functions use the MATLAB function `randn` to obtain $N(0,1)$ random disturbances and select adequate initial conditions by themselves. As a general practice, it is advisable to omit the first observations of the simulated sample.

Example 6.1 (Simulation). To obtain a realization with 200 observations of the model:

$$\begin{aligned} y_{1t} &= .9 + .3y_{1t-1} + a_{1t} \\ y_{2t} &= .7 + .4y_{1t-1} + a_{2t} - .8a_{2t-4} \end{aligned} \qquad V\begin{bmatrix} a_{1t} \\ a_{2t} \end{bmatrix} = \begin{bmatrix} 1 & .9 \\ .9 & 1 \end{bmatrix}$$

the following code can be used:

```
[theta, din, lab] = arma2thd([-.3 NaN; -.4 NaN],[],[], ...
                             [NaN NaN; NaN -.8], [1 .9; .9 1], 4, [.9;.7], 1);
% Generates the exogenous variable simulates the sample on omits the first values
u = ones(250,1);
y = simmod(theta, din, 250, u); y=y(51:250,:)
```

6.7. Smoothing

Fixed-interval smoothing (FIS) consists of obtaining the first and second-order moments of a random vector conditional to all the data in a sample. Among other uses, it has been applied to interpolate missing data, Kohn and Ansley (1986), to "clean" signals contaminated by noise, Kohn and Ansley (1987), to obtain the exact score function of the parameters, Koopman and Shephard (1992), to compute efficient estimates of time-varying parameters, Swamy and Tavlas (1995), to compute the unobservable components in structural time-series models, Harvey (1989), and to calculate the residuals of a state-space model, Kohn and Ansley (1989).

Casals *et al.* (2000a) derive an exact algorithm that can be applied to stationary, non-stationary and partially non-stationary systems. The implementation of the algorithm is the following:

Forward step: Propagate a KF($\mathbf{0}$, $\mathbf{0}$), see Section 6.1, and:

$$\tilde{\Phi}_{t+1} = (\Phi - K_t H)\tilde{\Phi}_t \quad \text{with} \quad \tilde{\Phi}_1 = I \tag{33}$$

$$X_t = H\tilde{\Phi}_t \tag{34}$$

$$W_t = W_{t-1} + X_t^T B_t^{-1} X_t \quad \text{with} \quad W_0 = 0 \tag{35}$$

$$w_t = w_{t-1} + X_t^T B_t^{-1} \tilde{z}_t \quad \text{with} \quad w_0 = 0 \tag{36}$$

Backward step: Propagate:

$$r_{t-1} = H^T B_t^{-1} \tilde{z}_t + \bar{\Phi}_t^T \quad \text{with } r_N = 0 \tag{37}$$

$$R_{t-1} = H^T B_t^{-1} H + \bar{\Phi}_t^T R_t \bar{\Phi}_t \quad \text{with} \quad R_N = 0 \tag{38}$$

$$\bar{\Phi}_t = \Phi - K_t H \tag{39}$$

$$V_{t|N} = (I - P_{t|t-1}R_{t-1})\bar{\Phi}_t \tag{40}$$

$$x_{t|N} = x_{t|t-1} + P_{t|t-1}r_{t-1} + V_{t|N}x_{1|N} \tag{41}$$

$$P_{t|N} = P_{t|t-1} - P_{t|t-1}R_{t-1}P_{t|t-1} + V_{t|N}P_{1|N}V_{t|N}^T \tag{42}$$

where:

$$x_{1|N} = P_{1|N}(P_1^{-1}\bar{x}_1 + w_N) \tag{43}$$

$$P_{1|N} = (P_1^{-1} + W_N)^{-1} \tag{44}$$

and the expressions $\bar{x}_1$ and P_1 in (43)-(44) are computed as described in Section 6.2. When the SS model has the innovation structure (5)-(6), the FIS can be further simplified to obtain both, computational an numerical stability advantages, see Casals *et al.* (2000a).

E⁴ implementation. The toolbox includes two basic FIS functions, `fismod` and `fismiss`. Their syntax is:

```
[xhat, Px, e] = fismod(theta, din, z);
[zhat, Pz, xhat, Px] = fismiss(theta, din, z);
```

where `fismiss` allows for missing values in `z` and the output arguments of `fismod` are `xhat`, the expectation of the state vector conditional on all the sample, see Eq. (41); `Px`, the covariance matrix of this expectation, see (42); and `e`, a matrix of FIS errors. On the other hand, `fismiss` has two additional arguments, `zhat`, which is a matrix containing the available values of `z` series and FIS estimates of its missing values and `Pz`, which is the covariance of `zhat`.

Example 6.2 (Forecasting and smoothing with generated data).

The model $z_t = (1 - .7B)(1 - .5B^{12})a_t$; $\sigma_a^2 = .1$, can be simulated and estimated with the following code:

```
[theta, din, lab] = arma2thd([], [], [-.7], [-.5], .1, 12);
% Compute the simulated sample and omit the first observations
z=simmod(theta,din,250); z=z(51:250,1);
% ... and then ML estimates
[thopt, it, lval, g, h] = e4min('lffast', theta,'', din, z);
[std, corrm, varm, Im] = imod(thopt, din, z);
prtest(thopt, din, lab, z, it, lval, g, h, std, corrm)
```

and the following code computes and plots ten out-of-the-sample forecasts, with the standard ±2σ limits:


```
[zfor, Bfor] = foremod(thopt, din, z, 10);
% The following is standard MATLAB code
figure; whitebg('w'); hold on
plot([z(191:200); zfor],'k-')
plot([z(191:200); zfor+2*sqrt(Bfor)],'k--')
plot([z(191:200); zfor-2*sqrt(Bfor)],'k--')
xlabel('Time')
hold off
```

Finally, some samples have missing values due to, *e.g.*, holidays or discontinuities in the source. Also, one may want to eliminate some observations because they are considered outliers and may affect the analysis. The following code generates a new variable with two missing values, interpolates them, using `fismiss` and displays the results:

```
z1=z; z1(10)=NaN; z1(40)=NaN;
[zhat, pz] = fismiss(thopt, din, z1);
[z(1:50) z1(1:50) zhat(1:50)]
```

7. Examples

7.1. Univariate modeling: the airline model

7.1.1. Estimating the airline model in stationary and nonstationary form

As it is well known, the multiplicative ARIMA$(0,1,1)\times(0,1,1)_{12}$ model describes adequately the logged airline data from Box, Jenkins and Reinsel (1994, pp. 547). The input code required to define and estimate the stationary version of the airline model is:

```
% Univariate model for the airline passenger series
load airline.dat
% Here begins E4 code.
e4init
z=transdif(airline,0,1,1,12);
[theta, din, lab] = arma2thd([],[],[0],[0],[0],12);
sete4opt('econd','zero','vcond','lyap','var','fac');
theta=e4preest(theta,din,z);
[thopt,it,lval,g,h]=e4min('lffast',theta,'',din,z);
[std, corrm, varm, Im] = imod(thopt,din,z);
prtest(thopt,din,lab,z,it,lval,g,h,std,corrm);
```

where the function `e4init` initializes the global toolbox options; `transdif` applies stationarity inducing transformations, see Terceiro *et al.* (2000), and the call to `sete4opt` is optional and allows to modify the toolbox options. In this case, it sets adequate initial conditions for the Kalman filter state and covariance matrix and chooses estimation in terms of the Cholesky factor of the error covariance, instead of the error covariance. As the model is scalar, this amounts to optimizing the likelihood with respect to the standard deviation of errors. The corresponding output is:

```
************** The following options are modified **************
Initial state vector. . . . . : ZERO
Initial covariance of state v. : LYAP
¿Variance or Cholesky factor?. : FAC
****************************************************************
```

The call to `e4preest` yields a fast and consistent preliminary estimation of the parameters in `theta`. This function uses a subspace-based algorithm, similar to generalized least-squares. Its estimates are adequate as final values when analyzing large samples or as starting values for maximum likelihood estimation of moderate size samples.

The E^4 function `e4min` implements a numerical optimization procedure, based on the techniques described by Dennis and Schnabel (1983). It supports two main optimization algorithms, BFGS (Broyden-Fletcher-Goldfarb-Shanno) and Newton-Raphson. MATLAB functions `fmin`, `fmins` and `fminu` can be used instead of `e4min`. However `e4min` has been carefully designed and tuned to solve likelihood optimization problems and, in most cases, it should be more reliable and robust for this specific use. In this example, `e4min` computes maximum likelihood estimates using numerical gradient, see Terceiro *et al.* (2000).

Finally, the call to `imod` produces analytical second-order moments for the estimates (under normality) and `prtest` summarizes and displays the estimation results:

```
******************** Results from model estimation ********************
  Objective function: -244.6965
     # of iterations:   12
Information criteria: AIC =  -3.6900, SBC =  -3.6242

Parameter          Estimate     Std. Dev.       t-test      Gradient
AR1(1,1)            -0.4018        0.0806      -4.9881        0.0000
AS1(1,1)            -0.5569        0.0846      -6.5850        0.0000
V(1,1)               0.0367        0.0023      16.1039        0.0000

************************* Correlation matrix **************************
AR1(1,1)       1.00
AS1(1,1)       0.00  1.00
V(1,1)         0.00  0.10  1.00

           Condition number =   1.2243
Reciprocal condition number =   0.8372
***********************************************************************
```

Note that `V(1,1)=0.0367` is the residual standard error, according to `'var','fac'` option previously selected. The default option (`'var','var'`) would provide an estimate of the variance. The following code illustrates the use of fixed-value constraints and nonstationary models by replicating previous results using the model in nonstationary form:

```
load airline.dat
y=log(airline);
e4init
% Defines the non-stationary version of airline model.
[theta,din,lab] = arma2thd([-1],[-1],[0],[0],[0],12);
% The AR parameters, corresponding to regular and seasonal differences
%    are constrained to remain at its present values
theta = [theta zeros(size(theta))];
theta(1,2) = 1; theta(2,2) = 1;
% Note that the 'vcond ' option is now 'idej', because the model is nonstationary
sete4opt('econd','zero','vcond','idej','var','fac');
theta = e4preest(theta,din,y);
[thopt,it,lval,g,h] = e4min('lffast', theta,'', din, y);
[std,corrm,varm,im] = imod(thopt, din, y);
prtest(thopt,din,lab,y,it,lval,g,h,std,corrm);
```


and the output is:

```
******************** Results from model estimation ********************
  Objective function: -232.7503
     # of iterations:   15
Information criteria: AIC =  -3.1910, SBC =  -3.1291
Parameter          Estimate     Std. Dev.       t-test      Gradient
FR1(1,1)      *     -1.0000        0.0000        0.0000        0.0000
FS1(1,1)      *     -1.0000        0.0000        0.0000        0.0000
AR1(1,1)            -0.4018        0.0766       -5.2438        0.0002
AS1(1,1)            -0.5569        0.0738       -7.5450        0.0001
V(1,1)               0.0367        0.0022       16.9706        0.0012
*
denotes constrained parameter

************************* Correlation matrix **************************
AR1(1,1)       1.00
AS1(1,1)       0.00  1.00
V(1,1)         0.00  0.00  1.00

           Condition number =   1.0001
Reciprocal condition number =   0.9999
***********************************************************************
```

These results are very similar to those of the stationary formulation, so both models can be considered equivalent. However the nonstationary formulation has practical advantages for many applications. For example, it is simpler to forecast the level of a variable instead of forecasting its differences and then rebuilding the level; also, when a sample contains some missing values differencing is inadequate, as it propagates the missing data; finally, intervention of impulse-type outliers in high-frequency data by tagging them as missing is computationally very effective, as it reduces the number of parameters to be estimated.

7.1.2. Unconditional and conditional forecasting of the airline data

To illustrate these advantages, we will compute twelve forecasts using the function `foremod`. We will also compute forecasts conditional to a yearly growth of 25% in the number of passengers. This kind of constraint may result from extra-sample information or may be simply a target value to be achieved. In this very common situation, constrained forecasts are effective intermediate targets useful to monitor progress towards the objective.

```
% Computes the unconditional forecasts
[yfor,Bfor]=foremod(thopt,din,y,12);
Bfor=sqrt(Bfor);

% Computes the conditional forecast and plots the results
N=size(airline,1);
yobj=log(airline(N,1)*1.25); % End year target
yext=[y; NaN*ones(11,1); yobj];
[yhat Bhat]=fismiss(thopt,din,yext);

% The following is standard MATLAB code
figure; whitebg('w'); hold on
plot([exp(y((N-23):N,1));yfor*NaN],'k-')
plot([y((N-23):N,1)*NaN;exp(yfor)],'k--');
plot([y((N-23):N,1)*NaN;exp(yhat(N+1:N+12))],'ko');
plot([y((N-23):N,1)*NaN;exp(yfor+1.96*Bfor)],'k-.');
plot([y((N-23):N,1)*NaN;exp(yfor-1.96*Bfor)],'k-.');
grid
hold off
```

and the resulting plot is:

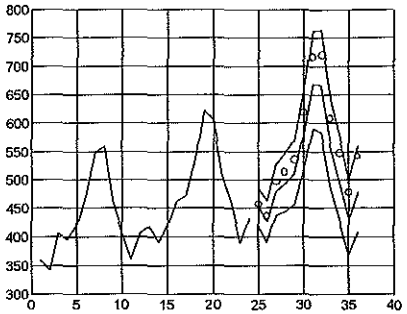

Figure 1: Conditional ('o') and unconditional forecasts ('-') for the airline passenger series.

7.2. Structural decomposition of a time series

7.2.1. A unobserved components time series model for the airline data.

As we said in Section 2.1, the airline model is very close to the following Structural Time Series Model (STSM):

$$y_t = T_t + S_t + e_t$$

$$T_{t+1} = T_t + \Delta_t$$

$$\Delta_{t+1} = \Delta_t + \eta_t$$

$$(1 + B + B^2 + B^3 + \ldots + B^{11})S_{t+1} = w_t$$

where the only parameters to be estimated are the variances in:

$$\mathrm{COV}\begin{bmatrix}\eta_t \\ w_t \\ e_t\end{bmatrix} = \begin{bmatrix}\sigma_\eta^2 & 0 & 0 \\ 0 & \sigma_w^2 & 0 \\ 0 & 0 & \sigma_e^2\end{bmatrix}$$

The code required to define this model in SS form and to obtain maximum-likelihood estimates for its parameters is:

```
% Structural time series model for the airline passenger series
load airline.dat
y=log(airline)*100;
e4init

% SS representation
Phi=zeros(13,13);
Phi(1,1)=1; Phi(1,2)=1; Phi(2,2)=1; Phi(3,13)=-1;
for i=3:12
   Phi(3,i)=-1;
   Phi(i+1,i)=1;
end
E=zeros(13,2); E(2,1)=1; E(3,2)=1;
H=zeros(1,13); H(1,1)=1; H(1,3)=1;
C=[1]; Q=[.01; 1.]; S=[0; 0]; R=[5];

% Defines the THD form
[theta,din,lab]=ss2thd(Phi, [], E, H, [], C, Q, S, R);

% All the parameters are constrained to zero except the error variances
theta=[theta ones(214,1)];
theta(210,2)=0; theta(211,2)=0; theta(214,2)=0;

% Chooses estimation of standard deviations, so negative estimates will not matter
%   and computes preliminary estimates
sete4opt('var','fac');
theta=e4preest(theta,din,y);

% ... and then ML estimates
[thopt,it,lval,g,h]=e4min('lffast',theta,'',din,y);
[std,corrm,varm,lm]=imod(thopt,din,y);
prtest(thopt,din,lab,y,it,lval,g,h,std,corrm);
```

and the resulting estimation output is:

```
******************** Results from model estimation ********************
  Objective function: 404.1350
      # of iterations:  36
Information criteria: AIC =   5.6547, SBC =   5.7165

Parameter          Estimate    Std. Dev.       t-test      Gradient
PHI(1,1)      *      1.0000       0.0000       0.0000       0.0000
PHI(2,1)      *      0.0000       0.0000       0.0000       0.0000
PHI(3,1)      *      0.0000       0.0000       0.0000       0.0000
PHI(4,1)      *      0.0000       0.0000       0.0000       0.0000

... [part of this listing has been deliberately omitted]

Q(1,1)                0.3841       0.0716       5.3650       0.0006
Q(2,2)                1.3688       0.2113       6.4767      -0.0002
S(1,1)        *       0.0000       0.0000       0.0000       0.0000
S(2,1)        *       0.0000       0.0000       0.0000       0.0000
R(1,1)                2.4591       0.2677       9.1855      -0.0001
* denotes constrained parameter

************************ Correlation matrix ************************
Q(1,1)         1.00
Q(2,2)         0.05  1.00
R(1,1)        -0.15 -0.48  1.00

          Condition number =   2.9636
Reciprocal condition number =   0.3332
************************************************************************
```

The final step in the analysis consist of estimating the components. The following code computes

and plots their FIS estimates:

```
% Computes FIS estimates of the structural components
% First, using the standard SS representation
[trend1,season1,cycle1,irreg1,theta1,din1] = ...
   e4trend(thopt,din,y,0);
plotsers([y trend1],1,str2mat('Data (log)','Trend'));
% As the SS model does not define the seasonal period, the seasonal component
%   is considered cyclic
plotsers(cycle1,-1,'Seasonal component');
plotsers(irreg1,-1,'Irregular component');
```

The input arguments for `e4trend` are a model in THD format, the values of the time series and a logical flag. Its current value ('0') conserves the SS structure of the model. The first output arguments provide FIS estimates of the trend, seasonal component, cycle and irregular component. There are more output arguments, not provided here, containing other results, see Terceiro *et al.* (2000). The resulting output is:

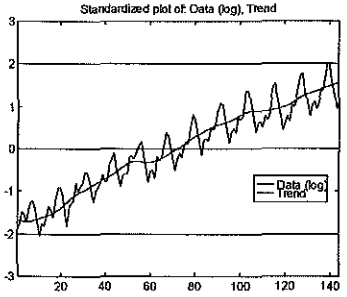

Figure 2

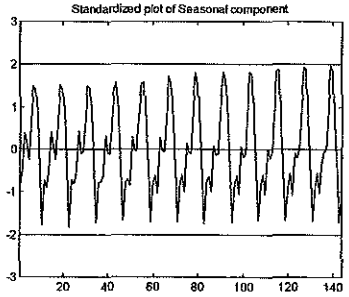

Figure 3

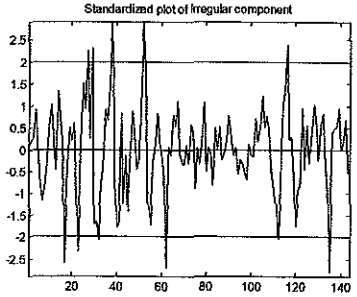

Figure 4

7.2.2. An exact model-based structural decomposition.

A different approach to compute the structural decomposition is described in Casals *et al.* (2000b). This approach only requires to transform the model for the data to the equivalent innovations representation and provides structural components which are unique, (conditionally) orthogonal and converge to exact values. E[4] computes this decomposition by changing the logical flag in the call to `e4trend` to the value '1':

```
% ... and now using the innovations form of the model
[trend2,season2,cycle2,irreg2,theta2,din2] = ...
   e4trend(thopt,din,y,1);
plotsers([y trend2],1,['Data log';'Trend   ']);
plotsers(cycle2,-1,'Seasonal component');
plotsers(irreg2,-1,'Irregular component');
```

and the resulting plots are:

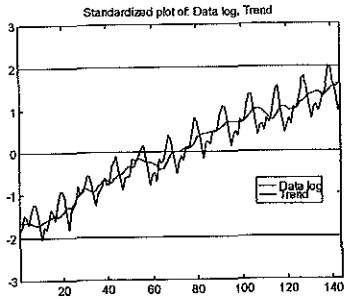

Figure 5

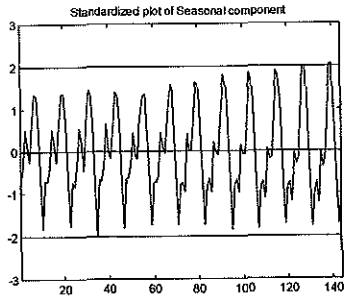

Figure 6

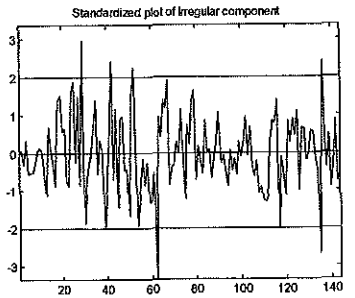

Figure 7

The trend and seasonal components obtained in both cases are very similar, but the irregular

components have different variances because the single error term in the innovations model is a combination of the three white noise variables in original model.

Despite their overall similarity, the properties of both sets of estimates are very different. Figures 8 and 9 compare the trace of the FIS covariances matrices of the states. Note that the variances of the STSM components display the "U-shape" characteristic of symmetric filters, meaning that the estimates at the center of the sample are more precise than those at the extremes. When the decomposition is applied to obtain seasonally adjusted data this fact is very important, because a change in the sample generates a revision effect proportional to the uncertainty of the components. In comparison, the components computed using the innovations model converge to values with null conditional variances and covariances.

The E^4 code required to compare both methods in this way is:

```
% Compare the results using the trace of the FIS covariance
[xhat1,px1]=fismod(theta1,din1,y);
[xhat2,px2]=fismod(theta2,din2,y);

trace1=[];trace2=[];
for i=1:13:size(px1,1)
   trace1=[trace1; trace(px1(i:i+12,:))];
   trace2=[trace2; trace(px2(i:i+12,:))];
end

plotsers([trace1 trace2],-1, ...
   str2mat('Trace (standard SS model)','Trace (innovations model)'));
```

and the resulting plots are:

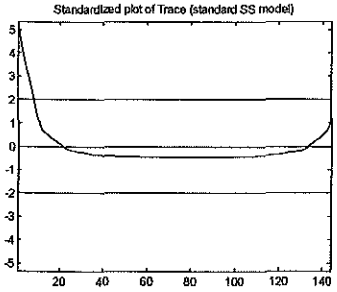

Figure 8

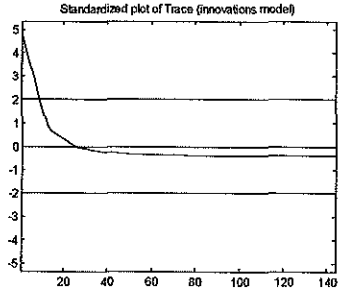

Figure 9

7.3. VARMA modeling: interaction between minks and muskrats

The number of muskrat (z_{1t}) and mink (z_{2t}) skins traded annually by the Hudson's Bay Company from 1848 to 1909 is a standard benchmark for multivariate methods, as the feedback interaction

between these series arises clearly from the fact that the mink is an important predator of the muskrat. After a cross-correlation analysis, Jenkins and Alavi (1981) propose a VARMA model for the relationship between both series:

$$\begin{bmatrix} 1+\phi_{11}^{1}B+\phi_{11}^{2}B^2+\phi_{11}^{3}B^3+\phi_{11}^{4}B^4 & 0 \\ 0 & 1+\phi_{22}^{1}B+\phi_{22}^{2}B^2 \end{bmatrix}\begin{bmatrix} \nabla \log z_{1t} \\ \log z_{2t}-\mu_2 \end{bmatrix}=\begin{bmatrix} 1+\theta_{11}^{1}B & \theta_{12}^{1}B \\ \theta_{21}^{1}B & 1+\theta_{22}^{1}B \end{bmatrix}\begin{bmatrix} a_{1t} \\ a_{2t} \end{bmatrix}$$

The following code defines and estimates this model:

```
% Bivariate modeling of the mink-muskrat series, Model (5.8) of Jenkins and Alavi
% (1981, pp. 37). The data is already in logs
load mink.dat;
e4init
z1=transdif(mink(:,3),1,1);
z2=mink(:,2)-mean(mink(:,2));
z=[z1 z2(2:62)];
% Define the parameter matrices and generate the THD representation
phi1 = [0 NaN; NaN   0]; phi2 = [0 NaN; NaN   0];
phi3 = [0 NaN; NaN NaN]; phi4 = [0 NaN; NaN NaN];
theta= [0   0;   0   0]; sigma= [0   0;   0   0];
[theta,din,lab]=arma2thd([phi1 phi2 phi3 phi4],[],[theta],[],sigma,1);
% Compute preliminary estimates
theta=e4preest(theta,din,z);
% Compute ML estimates, Information matrix and print the results
[thopt,it,l,g,H]=e4min('lffast',theta,'',din,z);
[std,corrm,varm,Im]=imod(thopt,din,z);
prtest(thopt,din,lab,z,it,l,g,H,std,corrm);
```

which yields:

```
******************** Results from model estimation ********************
  Objective function:   -9.0059
     # of iterations:    33
Information criteria: AIC =    0.1310, SBC =    0.5808

Parameter          Estimate     Std. Dev.      t-test      Gradient
FR1(1,1)           -0.6887        0.1364      -5.0510        0.0000
FR1(2,2)           -1.2680        0.1098     -11.5433        0.0000
FR2(1,1)            0.5941        0.1403       4.2346        0.0000
FR2(2,2)            0.5593        0.0921       6.0712        0.0000
FR3(1,1)           -0.0682        0.1171      -0.5821        0.0000
FR4(1,1)            0.2816        0.0856       3.2894        0.0000
AR1(1,1)           -0.2966        0.1606      -1.8468        0.0000
AR1(2,1)            0.6027        0.0804       7.4997        0.0000
AR1(1,2)           -0.8642        0.1387      -6.2299        0.0000
AR1(2,2)           -0.8352        0.1529      -5.4615        0.0000
V(1,1)              0.0639        0.0117       5.4854        0.0000
V(2,1)              0.0191        0.0072       2.6502        0.0000
V(2,2)              0.0423        0.0078       5.4491        0.0000

************************ Correlation matrix ************************
FR1(1,1)    1.00
FR1(2,2)   -0.02  1.00
FR2(1,1)   -0.68  0.34  1.00
FR2(2,2)   -0.23 -0.84 -0.15  1.00
FR3(1,1)    0.39 -0.19 -0.89  0.14  1.00
FR4(1,1)    0.35  0.15  0.35 -0.17 -0.65  1.00
AR1(1,1)    0.70 -0.36 -0.35  0.04  0.07  0.39  1.00
AR1(2,1)   -0.17  0.28  0.39 -0.35 -0.26  0.12  0.09  1.00
AR1(1,2)   -0.51  0.40  0.40 -0.38 -0.19 -0.22 -0.58  0.25  1.00
AR1(2,2)   -0.44  0.69  0.35 -0.45 -0.12 -0.21 -0.70  0.00  0.64  1.00
V(1,1)      0.00  0.03 -0.01 -0.03  0.01 -0.01  0.00 -0.02  0.02  0.02  0.02  1.00
V(2,1)     -0.01  0.02  0.00 -0.02  0.00 -0.01 -0.01  0.00  0.03  0.03  0.48  1.00
V(2,2)      0.01  0.01 -0.01 -0.02  0.01 -0.01  0.00 -0.02  0.02  0.02  0.12  0.48  1.00

        Condition number = 333.3643
Reciprocal condition number =   0.0025
************************************************************************
```

E[4] includes several functions for model validation and diagnosis. The following code computes the residuals of the VARMA specification, performs a descriptive multiple autocorrelation analysis and, finally, displays a standardized plot:

```
[ehat,vT,wT,vz1,vvT,vwT]=residual(thopt,din,z);
tit=str2mat('muskrat residuals','mink residuals');
descser(ehat,tit);
midents(ehat,10,tit);
plotsers(ehat,0,tit);
```

The residual descriptive statistics are:

```
*****************  Descriptive statistics  *****************

---  Statistics of muskrat residuals ---
Valid observations =    61
Mean               =    0.0150, t test =    0.4863
Standard deviation =    0.2403
Skewness           =    0.1883
Excess Kurtosis    =   -0.1468
Quartiles          =   -0.1481,   -0.0323,    0.1789
Minimum value      =   -0.5505, obs. #    58
Maximum value      =    0.6085, obs. #    26
Jarque-Bera        =    0.4152
Dickey-Fuller      =   -3.6353, computed with    7 lags
Dickey-Fuller      =   -8.2381, computed with    1 lags
Outliers list
 Obs #          Value
  26            0.6085
  37            0.5335
  52            0.4974
  56           -0.5056
  58           -0.5505

---  Statistics of mink residuals ---
Valid observations =    61
Mean               =   -0.0031, t test =   -0.1187
Standard deviation =    0.2033
Skewness           =   -0.4775
Excess Kurtosis    =    0.7708
Quartiles          =   -0.0937,    0.0061,    0.1055
Minimum value      =   -0.6718, obs. #    20
Maximum value      =    0.4089, obs. #    35
Jarque-Bera        =    3.8281
Dickey-Fuller      =   -4.1215, computed with    7 lags
Dickey-Fuller      =   -7.5664, computed with    1 lags
Outliers list
 Obs #          Value
  20           -0.6718
  30           -0.4616
  35            0.4089

Sample correlation matrix
    1.0000    0.3826
    0.3826    1.0000

Eigen structure of the correlation matrix
  i eigenval  %var | Eigen vectors
  1   1.3826 0.69 |   0.7071  0.7071
  2   0.6174 0.31 |   0.7071 -0.7071
************************************************************
```

The descriptive analysis indicates that the mean of both series is not statistically different of zero and the empirical distribution of the residuals is consistent with the normality assumption, as the skewness, excess kurtosis and Jarque-Bera statistics are small. Perhaps there are too many values exceeding two standard deviations from the mean that should be further investigated. On the other hand

the multiple autocorrelation analysis do not reveal any sign of misspecification except a high value of the Ljung-Box Q statistic in the (2,2) position, revealing that there could be some autocorrelation in the mink residuals:

```
******** Autocorrelation and partial autoregression functions ********
          MACF MPARF  MACF          MPARF
 k = 1, Chi(k) =    2.79, AIC(k) =  -6.08, SBC(k) =  -5.87
muskrat re  .. | .. |   -0.08  -0.07 | -0.06  -0.05
mink resid  .. | .. |    0.07   0.02 |  0.06  -0.01

 k = 2, Chi(k) =    4.77, AIC(k) =  -6.03, SBC(k) =  -5.69
muskrat re  .. | .. |   -0.14   0.03 | -0.18   0.11
mink resid  .. | -. |   -0.25  -0.12 | -0.21  -0.02

 k = 3, Chi(k) =    2.52, AIC(k) =  -5.95, SBC(k) =  -5.46
muskrat re  .. | .. |    0.17  -0.02 |  0.20  -0.13
mink resid  .. | .. |    0.01  -0.06 |  0.05  -0.09

 k = 4, Chi(k) =    3.47, AIC(k) =  -5.88, SBC(k) =  -5.26
muskrat re  .. | .. |   -0.01   0.09 | -0.01   0.17
mink resid  .. | .. |   -0.13  -0.04 | -0.18   0.05

 k = 5, Chi(k) =    4.11, AIC(k) =  -5.83, SBC(k) =  -5.07
muskrat re  .. | .. |   -0.05   0.16 | -0.11   0.19
mink resid  .. | .. |    0.06  -0.01 |  0.11  -0.11

 k = 6, Chi(k) =    7.83, AIC(k) =  -5.86, SBC(k) =  -4.96
muskrat re  .. | .. |   -0.16  -0.10 | -0.32   0.01
mink resid  .- | .- |   -0.19  -0.31 | -0.20  -0.22

 k = 7, Chi(k) =    6.18, AIC(k) =  -5.87, SBC(k) =  -4.83
muskrat re  .. | .. |   -0.08  -0.05 | -0.31   0.02
mink resid  .. | .. |    0.06  -0.10 |  0.15  -0.16

 k = 8, Chi(k) =   11.66, AIC(k) =  -6.00, SBC(k) =  -4.82
muskrat re  .. | .. |    0.07   0.01 |  0.02  -0.04
mink resid  .. | .. |   -0.10  -0.23 | -0.33  -0.22

 k = 9, Chi(k) =    1.99, AIC(k) =  -5.91, SBC(k) =  -4.60
muskrat re  .. | .. |    0.03   0.18 | -0.09   0.26
mink resid  .. | .. |    0.02   0.16 | -0.09   0.06

 k = 10, Chi(k) =    9.75, AIC(k) =  -6.02, SBC(k) =  -4.57
muskrat re  .. | .. |   -0.21  -0.13 | -0.41   0.03
mink resid  .+ | .+ |    0.21   0.36 | -0.15   0.27

The (i,j) element of the lag k matrix is the cross correlation (MACF)
or partial autoregression (MPACF) estimate when series j leads series i.

******************** Cross correlation functions ********************
lagged      muskrat re  mink resid
muskrat re  ..........  ..........
mink resid  ..........  .....-...+
Each row is the cross correlation function when column variable leads
row variable.

Ljung-Box Q statistic for previous cross-correlations
          muskrat re  mink resid
muskrat re    9.75        7.33
mink resid   12.48       23.85

Summary in terms of +.-
   For MACF std. deviations are computed as 1/T^0.5 =    0.13
   For MPARF std. deviations are computed from VAR(k) model
************************************************************************
```

Last, the residual plots are again quite satisfactory.

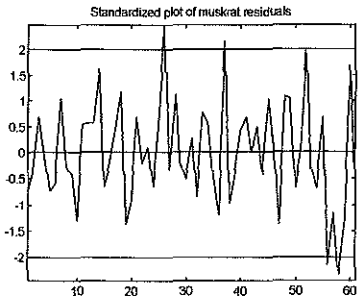

Figure 10

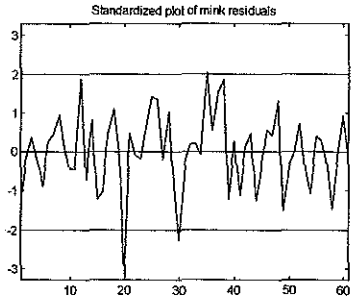

Figure 11

7.4. Conditional heteroscedastic modeling

In this application we measure short-run comovements of two currencies using the conditional correlations implied by a vector GARCH model. Consider the spot bid exchange rates of Deustche Mark (DM) and Japanese Yen (JY) against US Dollar, observed in the London Market during 695 weeks, from January 1985 to April 1998. The data has been logged, differenced and scaled by a factor of 100, to obtain the corresponding log percent yields. Excess returns are then computed by substracting the sample mean.

7.4.1. Univariate GARCH modeling

The first step of the analysis consists of building a model for the mean and the variance of these series. The code required to obtain the basic specification and descriptive statistics is the following:

```
load tiposc.dat
% The variables are sorted by columns in the following way:
% SP, CAD, FF, BP, SF, DM, IL, DF, JY

e4init;
% Choose DM/USD, transform the data and compute the excess returns
y=log(tiposc(:,6));
z=transdif(y,1,1)*100;
z=z-mean(z);

% Descriptive analysis of DM/USD
descser(z,'DM/USD excess returns');
uidents(z,20,'DM/USD excess returns');
uidents(z.^2,20,'DM/USD squared excess returns');
```

and the output is:


```
**********  Descriptive statistics  **********

---  Statistics of DM/USD excess returns ---
Valid observations =   694
Mean               =    0.0000, t test =    0.0000
Standard deviation =    1.3576
Skewness           =   -0.0460
Excess Kurtosis    =    1.6078
Quartiles          =   -0.8173,  -0.0479,    0.7775
Minimum value      =   -7.2574, obs. #   38
Maximum value      =    5.4025, obs. #  402
Jarque-Bera        =   74.9922
Dickey-Fuller      =   -4.9430, computed with  26 lags
Dickey-Fuller      =  -21.1610, computed with   1 lags
Outliers list
 Obs #         Value
  11          -2.7498

[Part of this listing was deliberately omitted]

 554           3.8374

************************************************************
```

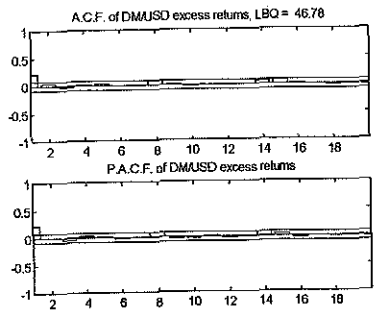

Figure 12

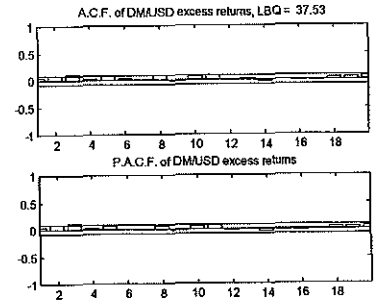

Figure 13

These results support an AR (1) process for the mean, see Figure 12, and a GARCH (1,1) model for the variance, see Figure 13. The code required to define and estimate this specification is:

```
% Model for the mean
[thetaf, dinf, labf]=arma2thd([0],[],[],[],[0],1);
% Model for the variance
[thetae, dine, labe]=arma2thd([0],[],[0],[],[0],1);
% Full model
[theta, din, lab] = garc2thd(thetaf, dinf, thetae, dine, labf, labe);

% Selects initial conditions of Kalman filter
sete4opt('econd','zero');

% Computes preliminary estimates
theta=e4preest(theta,din,z);
prtmod(theta,din,lab);
% ... and then ML estimates
[thopt,it,l,g,H]=e4min('lfgarch',theta,'', din,z);
[std,corrm,varm,Im]=igarch(thopt,din,z);
prtest(thopt,din,lab,z,it,l,g,H,std,corrm);
```

and the results are:

```
******************** Results from model estimation ********************
  Objective function: 1164.1261
     # of iterations:   19
Information criteria: AIC =    3.3664, SBC =    3.3925

Parameter          Estimate     Std. Dev.       t-test      Gradient
FR1(1,1)            -0.2185        0.0394      -5.5453       -0.0017
V(1,1)               1.8284        0.3039       6.0158        0.0000
FR1(1,1)            -0.9579        0.0233     -41.1184        0.0002
AR1(1,1)            -0.8745        0.0390     -22.4420       -0.0004

************************ Correlation matrix *************************
FR1(1,1)      1.00
V(1,1)        0.01  1.00
FR1(1,1)     -0.02 -0.34  1.00
AR1(1,1)     -0.01  0.05  0.80  1.00

         Condition number =  16.4800
Reciprocal condition number =   0.0545
*********************************************************************
```

As we said in Chapter 5, the representation employed for the GARCH part of the model is an ARMA for the squared errors. Therefore, these results can be written as:

$$z_t = \frac{\epsilon_t}{1 - .2185B}; \text{ with } \epsilon_t^2 = 1.8284 + N_t \;;\; (1 - .9579B)N_t = (1 - .8745B)v_t.$$

The following code: a) obtains residuals by calling the function `residual` (note that the last argument is equal to one, meaning that the residuals should be standardized), b) obtains basic diagnostics and, finally, c) plots the conditional volatility of this series:

```
[z1std,vT,wT,vz1] = residual(thopt,din,z,1);
plotsers(z1std,-1,'standardized residuals');
uidents(z1std,20,'standardized residuals');
uidents(z1std.^2,20,'squared standardized residuals');
% The following is standard MATLAB code
figure; whitebg('w'); close;
figure; hold on
axis([1 700 0 3]);
plot(sqrt(vz1),'k-');
title('conditional volatility of DM/USD excess returns');
hold off
```

and the resulting output is:

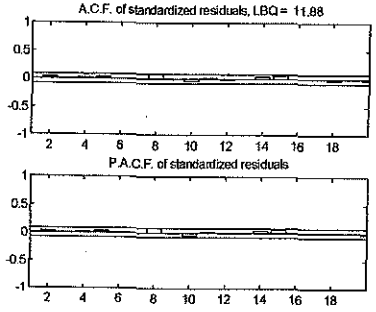

Figure 14

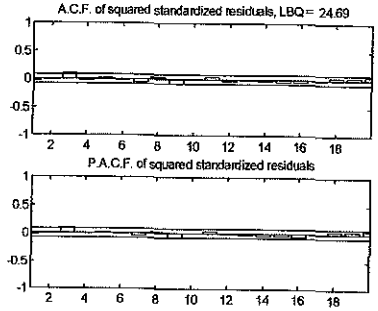

Figure 15

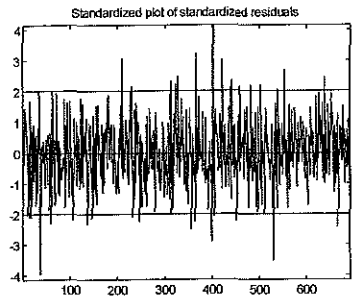

Figure 16

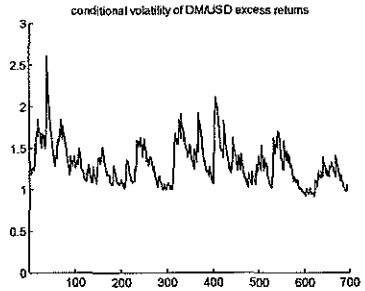

Figure 17

Using previous code it is easy to obtain analogous results for the JY series.

7.4.2. A multivariate GARCH model.

The literature has show interest in models with time-varying conditional covariances, but multivariate GARCH applications are scarce. This is due in part to difficulties in applying the gaussian maximum-likelihood algorithm to multivariate GARCH models. These difficulties have been implicitly justified by the fact that a general vector GARCH model has a large number of parameters, even for a moderate number of time series. Many authors suggest then simplifying assumptions like diagonal structure, Bollerslev *et al.* (1988), constant correlations, Bollerslev (1990) or factor ARCH structures, Engle (1987), Diebold and Nerlove (1989).

As Jerez *et al.* (1999) show, the likelihood of multivariate GARCH models is ill-conditioned because: a) the GARCH formulation induces flat likelihood surfaces in the surroundings of the maximum and b) when the series display high correlations - as often happens with financial data - the maximum is close to the zone of the parametric space where conditional covariances have negative eigenvalues. The interaction of these problems induce a extreme ill behavior of numerical algorithms.

The last step in our analysis consists of estimating a diagonal GARCH (1,1) model to the DM-JY excess returns. As the unconditional correlation of these series is relatively small (.64), the iterative algorithm is able to converge. The following code defines and estimates this model:

```
% Example: VAR(1)+VGARCH(1,1) for DM/USD-JY/USD excess returns
load tiposc.dat
% The variables are sorted in the following way:
% SP, CAD, FF, BP, SF, DM, IL, DF, JY
% Choose DM/USD and JY/USD, transform the data and compute the excess returns
e4init;
y=[log(tiposc(:,6)) log(tiposc(:,9))];
z=transdif(y,1,1)*100;
z(:,1)=z(:,1)-mean(z(:,1));
z(:,2)=z(:,2)-mean(z(:,2));
```

```
% The model is a diagonal VAR(1) for the mean
phi=[0 NaN;NaN 0];
sig=zeros(2);
[thetay,diny,laby]=arma2thd([phi],[],[],[],[sig],1);
% ... with a diagonal VGARCH(1,1) for the variance
phie1=[0 NaN NaN;NaN 0 NaN;NaN NaN 0];
thee1=[0 NaN NaN;NaN 0 NaN;NaN NaN 0];
[thetae, dine, labe]=arma2thd([phie1],[],[thee1],[],[zeros(3)],1);

% Compose both models and pre-estimate the parameters
[theta,din,lab]=garc2thd(thetay,diny,thetae,dine,laby,labe);
theta=e4preest(theta,din,z);

% Check that conditional covariances are positive-definite
[e2s,vT,wT,ve2] = residual(theta,din,z,1);
auval=[];
for t=1:2:size(ve2,1)
  eval=eig(ve2(t:t+1,:));
  auval=[auval;min(eval) max(eval)];
end
disp('*** Smallest eigenvalue:')
min(auval(:,1))

% Selects initial conditions of the Kalman filter
setе4opt('econd','zero');
%... and the ML estimates
[thopt,it,l,g,H]=e4min('lfgarch',theta,'', din,z);
[std,corrm,varm,Im]=igarch(thopt,din,z);
prtest(thopt,din,lab,z,it,l,g,H,std,corrm);
```

The resulting output is:

```
*** Smallest eigenvalue:

ans =

    0.0277

... [part of this listing has been deliberately omitted]

******************** Results from model estimation ********************
  Objective function: 2082.5467
     # of iterations:   49
Information criteria: AIC =   6.0333, SBC =   6.1053

Parameter          Estimate     Std. Dev.       t-test      Gradient
FR1(1,1)            -0.2444       0.0318      -7.6905        0.0016
FR1(2,2)            -0.3002       0.0340      -8.8158       -0.0007
V(1,1)               1.8096       0.2820       6.4166        0.0000
V(2,1)               1.1445       0.1553       7.3700        0.0003
V(2,2)               1.6656       0.1708       9.7519       -0.0001
FR1(1,1)            -0.9729       0.0120     -80.8656        0.0007
FR1(2,2)            -0.9420       0.0153     -61.4178        0.0057
FR1(3,3)            -0.8609       0.0350     -24.5948       -0.0030
AR1(1,1)            -0.9158       0.0206     -44.4318       -0.0006
AR1(2,2)            -0.8726       0.0239     -36.5462       -0.0060
AR1(3,3)            -0.7390       0.0468     -15.7981        0.0027

************************ Correlation matrix ************************
FR1(1,1)      1.00
FR1(2,2)      0.49  1.00
V(1,1)        0.02  0.01  1.00
V(2,1)        0.02  0.02  0.78  1.00
V(2,2)        0.01  0.02  0.44  0.77  1.00
FR1(1,1)     -0.03 -0.02 -0.27 -0.05 -0.01  1.00
FR1(2,2)     -0.04 -0.05 -0.22 -0.11 -0.10  0.57  1.00
FR1(3,3)     -0.02 -0.05 -0.18 -0.25 -0.29  0.18  0.57  1.00
AR1(1,1)     -0.02 -0.01  0.07  0.16  0.15  0.79  0.37  0.00  1.00
AR1(2,2)     -0.03 -0.04  0.03  0.14  0.19  0.55  0.80  0.24  0.63  1.00
AR1(3,3)     -0.01 -0.04 -0.09 -0.08  0.03  0.24  0.62  0.85  0.16  0.51  1.00

          Condition number = 153.6100
Reciprocal condition number =   0.0055
************************************************************************
```


The following code computes standardized residuals, conditional volatilities and conditional correlations:

```
[e2s,vT,wT,ve2] = residual(thopt,din,z,1);
var1=ve2(1:2:size(ve2,1),1);
var2=ve2(2:2:size(ve2,1),2);
cov12=ve2(2:2:size(ve2,1),1);
corr=cov12./(sqrt(var1).*sqrt(var2));
figure
hold on
axis([1 700 0 1]);
plot(corr,'k-');
title('Conditional correlation of DM/USD-JY/USD returns');
hold off
```

and the result is:

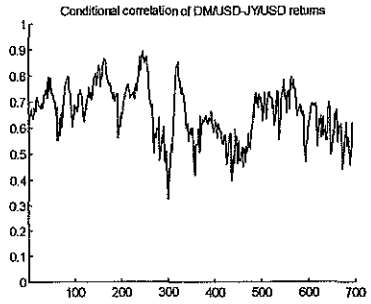

Figure 18

Note that the time-varying correlation fluctuates around its unconditional value (.64) with highs and lows of .9-.3 respectively. A high value of the conditional correlation means that the comovement of both series is strong in this period. On the contrary, low conditional correlations indicate periods when both currencies fluctuate independently.

8. Concluding remarks

In this paper we have discussed E^4, a MATLAB library of statistical and econometric algorithms for state-space models. We have also shown that a wide variety of models can be handled in this unified framework: from a simple ARIMA model to a GARCH multivariate model. The examples illustrate the enormous flexibility of this approach.

Next releases of E^4 will provide routines for subspace-based algorithms, see Van Overschee and de Moor (1996), Viberg (1995), Casals (1997), VARMAX echelon modeling, see Reinsel (1993) and specific support for new models such as periodic VARMAX, see Casals *et al.* (1998), state-space models with regime switching, see Kim and Nelson (1999) or multiple-input/ multiple-output transfer functions, see Box *et al.* (1994).

Acknowledgments

Sonia Sotoca gratefully acknowledges financial support from the CICYT project PB98-0789/98.

References

Anderson, B. D. O. and J. B. Moore (1979). *Optimal Filtering*. Englewood Cliffs (N.J.): Prentice Hall.

Bollerslev, T., R.F. Engle and J.M. Wooldridge (1988). "A Capital-Asset Pricing Model with Time-Varying Covariances", *Journal of Political Economy*, 96/1, 116-131.

Bollerslev, T. (1990). "Modeling the Coherence in Short-Run Nominal Exchange-Rates: A Multivariate Generalized ARCH Approach". *Review of Economics and Statistics*, 72, 498-505.

Bollerslev, T., R.F. Engle and D.B. Nelson (1994). "ARCH Models", in R.F. Engle and D.L. McFadden (editors), *Handbook of Econometrics*, vol. IV. Amsterdam: North-Holland.

Box, G.E.P., G. M. Jenkins and G.C. Reinsel (1994). *Time Series Analysis, Forecasting and Control*. Englewood Cliffs (N. J.): Prentice-Hall.

Casals, J. (1997). *Métodos de Subespacios en Econometría*. Phd Thesis. Madrid: Universidad Complutense.

Casals, J. and S. Sotoca (1997). "Exact Initial Conditions for Maximum Likelihood Estimation of State Space Models with Stochastic Inputs," *Economics Letters*, 57, 261-267.

Casals, J., S. Sotoca and M. Jerez (1998). "Un Algoritmo Rápido para Evaluar la Verosimilitud Exacta de Modelos VARMAX Periódicos". *Estadística Española*, 40, 143, 269-291.

Casals, J., S. Sotoca and M. Jerez (1999). "A Fast and Stable Method to Compute the Likelihood of Time Invariant State-Space Models," *Economics Letters*, 65, 3, 329-337.

Casals, J., M.. Jerez and S. Sotoca (2000a). "Exact Smoothing for Stationary and Nonstationary Time Series," *International Journal of Forecasting*, 16, 59-69 .

Casals, J., M. Jerez and S. Sotoca (2000b). "An Exact Multivariate Model-based Structural Decomposition". This paper can be downloaded from http://www.ucm.es/info/icae/e4.

Casals, J. and S. Sotoca (2000). "The Exact Likelihood for a State Space Model with Stochastic Inputs". *Computers and Mathematics with Applications* (forthcoming).

De Jong, P. (1988). "The Likelihood of a State Space Model". *Biometrika*, 75, 165-169.

De Jong, P. and S. Chu-Chun-Lin (1994). "Stationary and Non-Stationary State Space Models," *Journal of Time Series Analysis*, 15, 2, 151-166.

Dennis, J.E. and R.B. Schnabel (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Englewood Cliffs (N. J.): Prentice-Hall.

Diebold, F.X. and M. Nerlove (1989). The Dynamics of Exchange Rate Volatility: a Multivariate Latent Factor ARCH Model". *Journal of Applied Econometrics*, 4, 1-21.

Engle, R.F. (1982). "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of U.K. Inflation," *Econometrica*, 50, 987-1008.

Engle, R.F. (1984). "Wald, Likelihood and Lagrange Multiplier Tests in Econometrics", in Z. Griliches and M.D. Intriligator (editors), *Handbook of Econometrics*, vol. II. Amsterdam: North-Holland.

Engle, R.F. (1987). "Multivariate ARCH with Factor Structures-Cointegration in Variance. Discussion Paper 87-2, University of California, San Diego.

Harvey, A.C. (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge: Cambridge University Press.

Harvey, A.C., E. Ruiz and E. Sentana (1992). "Unobserved Component Time Series Models with ARCH Disturbances". *Journal of Econometrics*, 52, 129-157.

Hodrick, R.J. and E.C. Prescott (1980). "Post-war U.S. Business Cycles," *Carnegie Mellon University Working Paper*.

Jenkins, G.M. and A.S. Alavi (1981). "Some Aspects of Modelling and Forecasting Multivariate Time Series," *Journal of Time Series Analysis*, 2, 1, 1-47.

Jerez, M., J. Casals and S. Sotoca (1999). "The Likelihood of Multivariate GARCH Models is Ill-Conditioned". This paper can be downloaded from http://www.ucm.es/info/icae/e4.

Kim, C. and C.R. Nelson (1999). *State-Space Models with Regime Switching*. Cambridge: Massachusetts, The MIT Press.

Kohn, R. and C.F. Ansley (1986). "Estimation, Prediction and Interpolation for ARIMA Models with Missing Data. *Journal of the American Statistical Association*, 81, 751-761.

Kohn, R. and C.F. Ansley (1987). "Signal Extraction for Finite Nonstationary Time Series". *Biometrika*, 74, 411-421.

Kohn, R. and C.F. Ansley (1989). "A Fast Algorithm for Signal Extraction, Influence and Cross-Validation in State Space Models". *Biometrika*, 76, 65-79.

Koopman, S.J. and N. Shephard (1992). "Exact Score for Time Series Model in State Space Form". *Biometrika*, 79, 823-826.

MATLAB (1992). *MATLAB: Reference Guide*. Natick, Mass.: The MathWorks Inc.

MATLAB (1992). *MATLAB: External Interface Guide*. Natick, Mass.: The MathWorks Inc.

MATLAB (1996). *MATLAB COMPILER: Users Guide*. Natick (Mass): The MathWorks Inc.

McCullough, B.D. and H.D. Vinod (1999). "The Numerical Reliability of Econometric Software", *Journal of Economic Literature*, XXXVII, 633-665.

Reinsel, G.C. (1993). *Elements of Multivariate Time Series Analysis*. Berlin: Springer-Verlag.

Swamy, P.A.V.B. and G.S. Tavlas (1995). "Random Coefficients Models: Theory and Applications," *Journal of Economic Surveys*, 9, 2, 165-196.

Terceiro, J. (1990). *Estimation of Dynamic Econometric Models with Errors in Variables*. Berlin: Springer-Verlag.

Terceiro, J. (1999), "Comments on Kalman Filtering Methods for Computing Information Matrices for Time-Invariant Periodic and Generally Time-Varying VARMA Models and Samples", *Computers & Mathematics with Applications* (forthcoming).

Terceiro, J., J.M. Casals, M. Jerez, G.R. Serrano and S. Sotoca (2000). *Time Series Analysis using MATLAB. Including a complete MATLAB Toolbox*. This is the Reference Manual of E^4, and can be downloaded from http://www.ucm.es/info/icae/e4.

Van Overschee, P. and B. De Moor (1996). *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Dordretch: Kluwer Academic Publishers.

Viberg, M. (1995). "Subspace-based methods for the identification of linear time-invariant systems," *Automatica*, 31, 12, 1835-1851.

Watson, M.W. and R.F. Engle (1983). "Alternative Algorithms for the Estimation of Dynamic Factor, MIMIC and Varying Coefficient Regression Models," *Journal of Econometrics*, 23, 3, 385-400.

White, H. (1982). "Maximum Likelihood Estimation of Misspecified Models," *Econometrica*, 50, 1, 1-25.

APPENDIX: Summary of the main E[4] functions.

Model formulation	
*form*2THD	Where *form* may be ARMA, STR, TF, SS or GARC. Converts a VARMAX (ARMA), structural econometric model (STR), transfer function (TF), SS model (SS) or model with GARCH errors (GARC) to THD format.
STACKTHD	Stacks to models in THD format.
COMP2THD	Converts a stacked model in a components model in THD format.
NEST2HD	Converts a stacked model in a nested model in THD format.
THD2*form*	Where *form* may be ARMA, STR, TF or SS. Converts a model in THD format to the corresponding VARMAX, structural econometric, transfer function or SS formulation.

Model information	
PRTMOD	Displays information about a model.
PRTEST	Displays the estimation results.

Model estimation	
E4PREEST	Computes a fast estimate of the parameters for a model in THD form.
LFMOD	Computes the exact log-likelihood function for a model in THD form.
LFFAST	A faster version of LFMOD.
LFMISS	Same as LFMOD, but allowing for missing data.
LFGARCH	Same as LFMOD, but allowing for GARCH errors.
GMOD	Computes the analytical gradient of LFMOD.
GMISS	Computes the analytical gradient of LFMISS.
GGARCH	Computes the gradient of LFGARCH.
IMOD	Computes the exact information matrix of LFMOD and LFFAST.
IMODG	Computes the quasi-maximum likelihood information matrix of LFMOD and LFFAST.
IMISS	Computes the exact information matrix of LFMISS.
IGARCH	Computes an analytical approximation to the information matrix of LFGARCH.

Forecasting, smoothing and simulation	
FOREMOD	Computes forecasts for the endogenous variables of a model in THD form.
FOREMISS	Same as FOREMOD, but allowing for missing data.
FOREGARC	Computes forecasts for the endogenous variables and conditional variances of a model with GARCH errors.
FISMOD	Computes fixed-interval smoothing estimates of the state and observable variables of a model in THD form.
FISMISS	Same as FISMOD, but allowing for missing data.
SIMMOD	Simulates the endogenous variables of a model in THD form.
SIMGARCH	Same as SIMMOD, but allowing form GARCH errors.
E4TREND	Decomposes a vector of time series into trend, seasonal, cycle and irregular components.

Data transformation, model specification and diagnosis	
AUGDFT	Computes the augmented Dickey-Fuller test for unit roots.
DESCSER	Displays the main descriptive statistics for a set of time series.
HISTSERS	Displays a standardized histogram for a set of time series.
LAGSER	Generates lags and leads for a set of time series.
MIDENTS	Computes and displays the multiple autocorrelation and partial autoregression functions for a set of time series.
PLOTQQS	Plots the quantile graphs for a set of time series.
PLOTSERS	Displays a plot of centered and standardized time series versus time.
RESIDUAL	Computes the residuals of a model.
RMEDSER	Displays a scaled plot of sample means versus sample standard deviations for a set of time series.
TRANSDIF	Applies stationarity inducing transformations (Box-Cox and differencing) to a set of time series.
UIDENTS	Displays the univariate simple and partial autocorrelation functions for a set of time series.

Other functions	
E4INIT	Initializes the global toolbox options.
E4MIN	Computes the unconstrained minimum of a nonlinear function.
SETE4OPT	Allows the user to modify the toolbox options.